

DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD  
DDD DDD EEE BBB BBB UUU UUU GGG  
DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD  
DDDDDDDDDDDDDD

\*\*FILE\*\*ID\*\*DBGLEVEL1

J 16

DDDDDDDDDD BBBBBBBBBB GGGGGGGGGG LL EEEEEEEEEE VV VV EEEEEEEEEE LL 11  
DDDDDDDDDD BBBBBBBBBB GGGGGGGGGG LL EEEEEEEEEE VV VV EEEEEEEEEE LL 11  
DD DD BB BB GG LL EE VV VV EE LL 1111  
DD DD BB BB GG LL EE VV VV EE LL 1111  
DD DD BB BB GG LL EE VV VV EE LL 11  
DD DD BB BB GG LL EE VV VV EE LL 11  
DD DD BBBBBBBBBB GG LL EEEEEEEE VV VV EEEEEEEE LL 11  
DD DD BBBBBBBBBB GG LL EEEEEEEE VV VV EEEEEEEE LL 11  
DD DD BB BB GG GGGGGG LL EE VV VV EE LL 11  
DD DD BB BB GG GGGGGG LL EE VV VV EE LL 11  
DD DD BB BB GG GG LL EE VV VV EE LL 11  
DD DD BB BB GG GG LL EE VV VV EE LL 11  
DDDDDDDDDD BBBBBBBBBB GGGGGGGG LLLLLLLL EEEEEEEEEE VV VV EEEEEEEEEE LL 11111  
DDDDDDDDDD BBBBBBBBBB GGGGGGGG LLLLLLLL EEEEEEEEEE VV VV EEEEEEEEEE LL 11111

The image shows a 10x10 grid of binary symbols. The symbols are arranged to form a stylized arrow pointing to the right. The 'L' symbols are located in the top-left corner, while the 'S' symbols are in the top-right corner. The arrow's body is formed by a series of vertical and diagonal lines of 'I' symbols.

```
1 0001 0 MODULE DBGLEVEL1 (IDENT = 'V04-000') =
2 0002 1 BEGIN
3 0003 1 ++
4 0004 1 ****
5 0005 1 *
6 0006 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7 0007 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8 0008 1 * ALL RIGHTS RESERVED.
9 0009 1 *
10 0010 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 0011 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 0012 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 0013 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 0014 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 0015 1 * TRANSFERRED.
16 0016 1 *
17 0017 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 0018 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 0019 1 * CORPORATION.
20 0020 1 *
21 0021 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 0022 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 0023 1 *
24 0024 1 *
25 0025 1 ****
26 0026 1 --
27 0027 1 ++
28 0028 1 FACILITY: DEBUG (DBG)
29 0029 1 ABSTRACT:
30 0030 1 This module contains all the miscellaneous routines left over from
31 0031 1 the early days of the debugger. That is, the debugger was mostly
32 0032 1 rewritten in 1982-1983 but after the rewrite there were still
33 0033 1 a handful of routines in different modules that were still used.
34 0034 1 These routines have all been lumped together in this one module.
35 0035 1
36 0036 1
37 0037 1 Version: 4.0
38 0038 1
39 0039 1 History:
40 0040 1 Created by:
41 0041 1 R. Title , May 1983
42 0042 1
43 0043 1
44 0044 1
45 0045 1 Require files:
46 0046 1
47 0047 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
48 0181 1 LIBRARY 'LIB$:DBGGEN.L32';
49 0182 1
50 0183 1 Table of contents:
51 0184 1
52 0185 1 FORWARD ROUTINE
53 0186 1 dbgSend_of_cmd : NOVALUE,
54 0187 1 dbgSend_of_line : NOVALUE,
55 0188 1 dbg$write_mem,
56 0189 1 dbg$set_context: NOVALUE,
57 0190 1 dbg$init_debug : NOVALUE,
```

! END OF COMMAND PROCESSING ROUTINE  
! end of line processing routine  
! WRITES data TO MEMORY  
! ROUTINE TO INITIALIZE CONTEXT BITS  
! ROUTINE TO INITIALIZE DEBUG UNDER STARLET

```

58      0191 1    dbgScis_connectcf : NOVALUE,
59      0192 1    dbgScis_remove : NOVALUE,
60      0193 1    dbgScis_add : NOVALUE,
61      0194 1    dbgSins_opcodes: NOVALUE,
62      0195 1    dbg$conv_r 50,
63      0196 1    dbgSout_Regname,
64      0197 1    dbgSreg_match,
65      0198 1    dbgSdigit_scan,
66      0199 1    dbg$output_psl : NOVALUE,
67      0200 1    dbg$map_to_reg_addr,
68      0201 1
69      0202 1    dbg$exact_map_to_reg,
70      0203 1
71      0204 1    DBGSLANGUAGE,
72      0205 1    DBGSSET_LANG;
73      0206 1
74      0207 1 ! Externals
75      0208 1
76      0209 1 EXTERNAL ROUTINE
77      0210 1    dbg$ins_decode,
78      0211 1
79      0212 1    dbg$ins_encode,
80      0213 1    dbg$newline : NOVALUE,
81      0214 1    dbg$pop_tempmem: NOVALUE,
82      0215 1    dbg$push_tempmem,
83      0216 1    dbg$sta_getsourcmod,
84      0217 1
85      0218 1    dbg$src_type_lnum_source : NOVALUE,
86      0219 1    dbg$src_type_pc_source : NOVALUE,
87      0220 1
88      0221 1    dbg$def_pr_entry,
89      0222 1    dbg$get_tempmem,
90      0223 1    dbg$ncis_add,
91      0224 1
92      0225 1    DBGSCANCEL_LOC_VAL: NOVALUE, ! Cancels '.' and '\''
93      0226 1    dbg$check_prot,
94      0227 1    DBGSEVENT_INITIALIZATION : NOVALUE,
95      0228 1    dbg$fao_out: NOVALUE,
96      0229 1    DBGSFLUSHBUF: NOVALUE,
97      0230 1    dbg$get_memory,
98      0231 1    dbg$init_define: NOVALUE,
99      0232 1    dbg$init_memory: NOVALUE,
100     0233 1    dbg$init_modes,
101     0234 1    dbg$init_search: NOVALUE,
102     0235 1    DBGSNCANCEL_LOC AND VAL : NOVALUE,
103     0236 1    DBGSNCHANGE_TO_NEW : NOVALUE, ! Switch to new debugger
104     0237 1    dbg$ncis_remove,
105     0238 1    dbg$nget_trans_radix,
106     0239 1    DBGSPARSER_SET_LANGUAGE : NOVALUE,
107     0240 1    dbg$print : NOVALUE,
108     0241 1    DBGSPRINT_CONTROL,
109     0242 1    dbg$read_access,
110     0243 1    dbg$redo_prot,
111     0244 1    dbg$rel_memory : NOVALUE,
112     0245 1    DBGSREL_TEMPMEM: NOVALUE,
113     0246 1    DBGSRST_TEMP_RELEASE: NOVALUE,
114     0247 1    dbg$set_define_def: NOVALUE.

    | Places fcf into input stream
    | Remove a link from the cis
    | Add a link to the cis
    | SPECIAL-PURPOSE RAD50 CONVERSION ROUTINE.
    | Match and symbolize register names
    | MATCHES A STRING TO A REGISTER NAME
    | scan string for valid numeric
    | ROUTINE TO OUTPUT PSL IN SPECIAL FORMAT
    | Tries to map input address to an address
    | in the reg area in user_runframe
    | Tries to map input address to the address
    | of a reg in user runframe
    | Produce name of given language.
    | Changes the DEBUG syntax

    | ROUTINE TO OUTPUT MEMORY AS
    | SYMBOLIC INSTRUCTIONS.
    | routine to encode a symbolic instruction
    | ACTUALLY DO TERMINAL I/O.
    | Pop a temporary memory pool
    | Create a new temporary memory pool
    | looks up module rst pointer
    | given the RST pointer
    | types a range of source line nums
    | types source for a range
    | of PCs.
    | Procedure entry code
    | allocate temporary memory
    | add a CIS to the cis chain

    | Cancels '.' and '\''
    | CHECKS THE PROTECTION OF A PAGE
    | Initialize event structures

    | Flush the print buffer
    | Allocate permanent memory
    | Initializes define settings
    | Initialize the free memory pool
    | INITIALIZES MODES
    | Initializes search settings
    | Cancels '.', and '\''

    | Translate radix
    | Set up parse table for new language
    | FORMATTED BUFFERED OUTPUT.
    | Set print control functions
    | verify access to memory
    | RESETS THE PROTECTION OF A PAGE TO READ ONLY
    | Release memory
    | Release all temporary memory
    | Release temporary RST entries
    | Initializes DEFINE data struc

```

115	0248	1	dbg\$set_define_lvl: NOVALUE,	Sets define level back
116	0249	1	DBGSET_MOD_DEF.	initializes mode settings
117	0250	1	dbg\$set_mod_lvl.	SETS MODE pointer
118	0251	1	dbg\$set_out_def : NOVALUE,	Initializes OUTPUT config.
119	0252	1	DBGSET_SEARCH_DEF : NOVALUE,	! initialize search settings
120	0253	1	dbg\$set_search_lvl: NOVALUE,	Sets search level back
121	0254	1	DBGSET_STP_DEF,	Initializes step settings
122	0255	1	dbg\$sta_setcontext : NOVALUE,	Sets registers context
123	0256	1	dbg\$sta_symname : NOVALUE,	Get symbol's name
124	0257	1	forScnv_in_defg,	converts a floating or real
125	0258	1	smg\$create_key_table,	/Initialize data structures
126	0259	1	smg\$create_virtual_keyboard,	\ used for keypad input.
127	0260	1	sysStrnlog,	Translate logical name
128	0261	1	dbg\$sta_symvalue : NOVALUE,	
129	0262	1	dbg\$free_mem_left,	! Longwords remaining in free storage.
130	0263	1	dbg\$make_arg_vect,	
131	0264	1	dbg\$out_info,	
132	0265	1	forScnv_out_i,	! Converts integer to ascii string.
133	0266	1	dbg\$npaPathdesc_to_cs : NOVALUE,	Get full name of data item
134	0267	1	lib\$get_ef,	Get event flag
135	0268	1	lib\$free_ef;	Free event flag
136	0269	1		
137	0270	1	EXTERNAL	
138	0271	1	dbg\$gb_set_break_flag: BYTE,	Flag set to true when parsing
139	0272	1		a SET BREAK command.
140	0273	1	dbg\$gb_radix: VECTOR[3,BYTE],	Radix settings
141	0274	1	dbg\$gl_context: BITVECTOR,	CONTEXT WORD
142	0275	1	dbg\$gl_developer: BITVECTOR,	Developer flags
143	0276	1	dbg\$gl_ind_com_file: REF VECTOR[,BYTE]	Points to counted string with
144	0277	1		indirect command file name
145	0278	1	dbg\$gl_inpfab: BLOCK [, BYTE],	FAB FOR 'INPUT'
146	0279	1	dbg\$gl_inprab: BLOCK [, BYTE],	RAB FOR 'INPUT'
147	0280	1	dbg\$gl_outpfab: BLOCK [, BYTE],	FAB FOR 'OUTPUT'
148	0281	1	dbg\$gl_outprab: BLOCK [, BYTE],	RAB FOR 'OUTPUT'
149	0282	1	dbg\$gl_symhead,	LIST HEAD FOR SYMBOL TABLE
150	0283	1	dbg\$gl_global_define_ptr,	Head of DEFINE list for
151	0284	1		globally defined symbols
152	0285	1	dbg\$gl_local_define_ptr,	Head of DEFINE list for
153	0286	1		locally defined symbols
154	0287	1		
155	0288	1	dbg\$gl_lis_ptr,	Used for DEFINE/KEY
156	0289	1	dbg\$gl_key_table_id,	Used for DEFINE/KEY
157	0290	1	dbg\$gl_keyboard_id,	TRUE if we are trying to do
158	0291	1	dbg\$gb_keypad_input: BYTE,	keypad input.
159	0292	1	dbg\$src_term_width,	Terminal set width
160	0293	1	dbg\$gb_exc_bre_flag: BYTE,	TRUE during an exception break
161	0294	1	dbg\$gb_go_arg_flag: BYTE,	TRUE if there is an argument
162	0295	1		to GO.
163	0296	1	dbg\$gl_help_input,	Pointer to HELP input
164	0297	1	dbg\$gb_search_ptr: REF VECTOR[,BYTE],	Pointer to search structure
165	0298	1	dbg\$gb_mod_ptr: REF VECTOR [,BYTE],	Pointer to the mode structure
166	0299	1	dbg\$gb_set_module_flag: BYTE,	TRUE during SET MODULE command.
167	0300	1	dbg\$gb_resignal: BYTE,	FLAG FOR RESIGNALING EXCEPTIONS
168	0301	1	dbg\$gb_take_cmd: BYTE,	FLAG WHICH SAYS CONTINUE TO ACCEPT COMMANDS
169	0302	1	dbg\$gw_locLength: word,	Length field of command override type
170	0303	1	dbg\$gl_dimenlst : VECTOR,	FORTRAN dimension list
171	0304	1	dbg\$gl_nest_level,	Nesting level of subscripts

```

172      0305 1   dbg$gl_nest_stack: VECTOR,
173      0306 1   dbg$gl_search_verb,
174      0307 1
175      0308 1   dbg$gl_set_source,
176      0309 1   dbg$gl_set_source2,
177      0310 1   dbg$gl_current_primary,
178      0311 1   dbg$gl_list: VECTOR,
179      0312 1   dbg$gl_loctyp,
180      0313 1   dbg$gl_dfltyp,
181      0314 1   dbg$gl_gbltyp,
182      0315 1   dbg$gl_stk : semantic_stack,
183      0316 1   DBG$GB_STP_PTR : REF EVENT$STEPPING_DESCRIPTOR,
184      0317 1   dbg$gl_step_num,
185      0318 1   dbg$pseudo_prog,
186      0319 1   dbg$gl_logTab : BLOCK [,BYTE],
187      0320 1   dbg$gl_cishead : REF cis$link,
188      0321 1   dbg$gl_modrstptr2,
189      0322 1
190      0323 1   dbg$gl_module,
191      0324 1
192      0325 1   dbg$gl_dirlist,
193      0326 1
194      0327 1
195      0328 1
196      0329 1   dbg$src_left_margin,
197      0330 1   dbg$src_right_margin,
198      0331 1   dbg$gv_control : dbg$control_flags,
199      0332 1   dbg$gw_gbllngth : WORD,
200      0333 1   dbg$gl_next_loc,
201      0334 1   dbg$gb_language : BYTE,
202      0335 1   dbg$reg_values : VECTOR,
203      0336 1   dbg$runframe : BLOCK [,BYTE],
204      0337 1   dbg$src_next_modrstptr,
205      0338 1
206      0339 1   dbg$src_next_lnum,
207      0340 1
208      0341 1
209      0342 1
210      0343 1   dbg$src_next_stmt,
211      0344 1   DBG$GL_GET_LEX,
212      0345 1   DBG$GL_PARTBPTR : VECTOR,
213      0346 1   DBG$GL_REDUC_RT,
214      0347 1   dbg$gb_def_out : VECTOR [,BYTE],
215      0348 1   dbg$gw_dflEng : WORD,
216      0349 1   rst$start_addr: REF rst$entry,
217      0350 1   dbg$gl_ascii_len,
218      0351 1   dbg$gb_loc_type: BYTE,
219      0352 1   dbg$gl_csp_ptr,
220      0353 1   dbg$gl_last_loc,
221      0354 1   dbg$gl_last_val;
222      0355 1
223      0356 1   ! Link symbol saying whether we are linking a debugger to run on a
224      0357 1   version 38 system.
225      0358 1
226      0359 1   EXTERNAL LITERAL
227      0360 1   dbg$gl_3b_system: WEAK;

```

| Stack of saved subscripts  
| Head of command execution tree  
| for SEARCH

| The primary being processed  
| LIST FOR EXPRESSIONS  
| command override type.  
| type given in SET TYPE.  
| type given in SET TYPE/OVERRIDE.  
| semantic stack for tokens etc.  
| DBG\$GB\_STP\_PTR : ! POINTER TO CURRENT STEP TYPE  
| number of steps to take in single step mode  
| Used for DEBUG's CALL command.  
| FAB for LOG file  
| Head of command input stream  
| Holds module pointer during  
| TYPE command.  
| Hold module pointer during  
| SET SOURCE/MODULE= command.  
| Holds pointer to directory  
| list during  
| SET SOURCE dir-list  
| command.  
| left margin for source display  
| right margin for source display  
| DEBUG control bits  
| OVERRIDE LENGTH  
| NEXT location TO DISPLAY  
| LANGUAGE INDEX  
| Context regs save area  
| current run frame  
| module pointer used by  
| dbg\$type\_cmd.  
| Contains next line num to  
| typed if no line num is  
| specified in the TYPE  
| command.  
| as above with stmt num  
| Holds name of current get lex routine  
| List of parse table addresses  
| Name of action routine for a syntax  
| Current OUTPUT configuration  
| The length specified in a SET TYPE statement.  
| Pointer to the module chain (MC).  
| Length of ascii string.  
| TYPE OF LAST LOCATION EXAMINED  
| pointer to current scope  
| CURRENT LOCATION  
| ! CURRENT VALUE

```

229      0361 1 GLOBAL ROUTINE DBGSNCOB_PATHDESC_TO_CS(pathname,name_string) : NOVALUE =
230      0362 2 BEGIN
231      0363 2     MAP pathname      : REF pth$pathname;
232      0364 2
233      0365 2     LOCAL
234      0366 2       name_vector   : REF VECTOR[,LONG];
235      0367 2       name_count,
236      0368 2       top_name      : REF VECTOR[,BYTE];
237      0369 2       sub_name      : REF VECTOR[,BYTE];
238      0370 2       pointer,length;
239      0371 2
240      0372 2       name_vector = pathname[pth$a_pathvector];
241      0373 2       name_vector = name_vector[.pathname[pth$b_pathcnt]];
242      0374 2       name_count = .pathname[pth$b_totcnt] - .pathname[pth$b_pathcnt];
243      0375 2       pathname[pth$b_totcnt] = .pathname[pth$b_pathcnt];
244      0376 2       dbg$ncob_pathdesc_to_cs(.pathname,top_name);
245      0377 2       length = .(top_name)<0,8,0>;
246      0378 2       DECR index FROM .name_count-1 TO 0 DO
247      0379 3         BEGIN
248      0380 3           sub_name = .name_vector[.index];
249      0381 3           IF .(sub_name)<0,8,0> GTR 0 THEN length= length+.sub_name)<0,8,0>+4;
250      0382 2         END;
251      0383 2         .name_string = pointer = dbg$get_tempmem(.length/%UPVAL)+1;
252      0384 2         ch$wchar_a(.length,pointer);
253      0385 2         DECR index FROM .name_count-1 TO 0 DO
254      0386 3           BEGIN
255      0387 3             sub_name = .name_vector[.index];
256      0388 3             IF .(sub_name)<0,8,0> GTR 0
257      0389 3             THEN
258      0390 4               BEGIN
259      0391 4                 ch$move(.(sub_name)<0,8,0>,sub_name[1],.pointer);
260      0392 4                 pointer = .pointer + .(sub_name)<0,8,0>;
261      0393 4                 ch$move(4,UPLIT BYTE(: of 'J,.pointer);
262      0394 4                 pointer = .pointer + 4;
263      0395 3               END;
264      0396 2             END;
265      0397 2             ch$move(.(top_name)<0,8,0>,top_name[1],.pointer);
266      0398 1             ! end of routine dbgsncob_pathdesc_to_cs

```

```

.TITLE DBGLEVEL1
.IDENT 'V04-000'
.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
20 66 6F 20 00000 P.AAA: .ASCII '\ of \
.EXTRN DBGSINS_DECODE, DBGSINS_ENCODE
.EXTRN DBGSNEWLINE, DBGSPOP_TEMPMEM
.EXTRN DBGSPUSH_TEMPMEM
.EXTRN DBGSSTA_GETSOURCEMOD
.EXTRN DBGSRC_TYPE_LNUM_SOURCE
.EXTRN DBGSRC_TYPE_PC_SOURCE
.EXTRN DBGSDEF_PR_ENTRY
.EXTRN DBGSGET_TEMPMEM
.EXTRN DBGSNCIS_ADD, DBGSCANCEL_LOC_VAL
.EXTRN DBGSCHECK_PROT, DBGSEVENT_INITIALIZATION
:
```

.EXTRN DBGSFAO\_OUT, DBGSFLUSHBUF  
.EXTRN DBGSGET\_MEMORY, DBGSINIT\_DEFINE  
.EXTRN DBGSINIT\_MEMORY  
.EXTRN DBGSINIT\_MODES, DBGSINIT\_SEARCH  
.EXTRN DBGSNCANCEL\_LOC\_AND\_VAL  
.EXTRN DBGSNCHANGE\_TO\_NEW  
.EXTRN DBGSNCIS\_REMOVE  
.EXTRN DBGSNGET\_TRANS\_RADIX  
.EXTRN DBGSPARSER\_SET\_LANGUAGE  
.EXTRN DBGSPRINT, DBGSPRINT\_CONTROL  
.EXTRN DBGSREAD\_ACCESS  
.EXTRN DBGSREDO\_PROT, DBGSREL\_MEMORY  
.EXTRN DBGSREL\_TEMPMEM  
.EXTRN DBGSRST\_TEMP\_RELEASE  
.EXTRN DBGSSET\_DEFINE\_DEF  
.EXTRN DBGSSET\_DEFINE\_LVL  
.EXTRN DBGSSET\_MOD\_DEF  
.EXTRN DBGSSET\_MOD\_LVL  
.EXTRN DBGSSET\_OUT\_DEF  
.EXTRN DBGSSET\_SEARCH\_DEF  
.EXTRN DBGSSET\_SEARCH\_LVL  
.EXTRN DBGSSET\_STP\_DEF  
.EXTRN DBGSSTA\_SETCONTEXT  
.EXTRN DBGSSTA\_SYMNAME  
.EXTRN FORSCNV\_IN\_DEFG  
.EXTRN SMG\$CREATE\_KEY\_TABLE  
.EXTRN SMG\$CREATE\_VIRTUAL\_KEYBOARD  
.EXTRN SYS\$TRNLOG, DBGSSTA\_SYMVALUE  
.EXTRN DBGSFREE\_MEM\_LEFT  
.EXTRN DBGSMAKE\_ARG\_VECT  
.EXTRN DBGSNOUT\_INFO, FORSCNV\_OUT\_I  
.EXTRN DBGSNPATDESC\_TO\_CS  
.EXTRN LIBSGET\_EF, LIBSFREE\_EF  
.EXTRN DBG\$GB\_SET\_BREAK\_FLAG  
.EXTRN DBG\$GB\_RADIX, DBGSGL\_CONTEXT  
.EXTRN DBGSGL\_DEVELOPER  
.EXTRN DBGSGL\_IND\_COM\_FILE  
.EXTRN DBGSGL\_INPFAB, DBGSGL\_INPRAB  
.EXTRN DBGSGL\_OUTPFAB, DBGSGL\_OUTPRAB  
.EXTRN DBGSGL\_SYMHEAD, DBGSGL\_GLOBAL\_DEFINE\_PTR  
.EXTRN DBGSGL\_LOCAL\_DEFINE\_PTR  
.EXTRN DBGSGL\_LIS\_PTR, DBGSGL\_KEY\_TABLE\_ID  
.EXTRN DBGSGL\_KEYBOARD\_ID  
.EXTRN DBGSGB\_KEYPAD\_INPUT  
.EXTRN DBGSRC\_TERM\_WIDTH  
.EXTRN DBGSGB\_EXC\_BRE\_FLAG  
.EXTRN DBGSGB\_GO\_ARG\_FLAG  
.EXTRN DBGSGL\_HELP\_INPUT  
.EXTRN DBGSGB\_SEARCH\_PTR  
.EXTRN DBGSGB\_MOD\_PTR, DBGSGB\_SET\_MODULE\_FLAG  
.EXTRN DBGSGB\_RESET\_SIGNAL  
.EXTRN DBGSGB\_TAKE\_CMD  
.EXTRN DBGSGW\_LOCLNGTH  
.EXTRN DBGSGL\_DIMENLST  
.EXTRN DBGSGL\_NEST\_LEVEL  
.EXTRN DBGSGL\_NEST\_STACK  
.EXTRN DBGSGL\_SEARCH\_VERB

		.EXTRN	DBG\$GL_SET_SOURCE		
		.EXTRN	DBG\$GL_SET_SOURCE2		
		.EXTRN	DBG\$GL_CURRENT_PRIMARY		
		.EXTRN	DBG\$GL_LIST, DBG\$GL_LOCTYP		
		.EXTRN	DBG\$GL_DFLTYP, DBG\$GL_GBLTYP		
		.EXTRN	DBG\$GL_STK, DBG\$GB_STP_PTR		
		.EXTRN	DBG\$GL_STEP_NUM		
		.EXTRN	DBG\$PSEUDO_PROG		
		.EXTRN	DBG\$GL_LOGFAB, DBG\$GL_CISHEAD		
		.EXTRN	DBG\$GL_MODRSTPTR2		
		.EXTRN	DBG\$GL_MODULE, DBG\$GL_DIRLIST		
		.EXTRN	DBG\$SRC_LEFT_MARGIN		
		.EXTRN	DBG\$SRC_RIGHT_MARGIN		
		.EXTRN	DBG\$GV_CONTROL, DBG\$GW_GBLNGTH		
		.EXTRN	DBG\$GL_NEXT_LOC		
		.EXTRN	DBG\$GB_LANGUAGE		
		.EXTRN	DBG\$REG_VALUES, DBG\$RUNFRAME		
		.EXTRN	DBG\$SRC_NEXT_MDRSTPTR		
		.EXTRN	DBG\$SRC_NEXT_LNUM		
		.EXTRN	DBG\$SRC_NEXT_STMT		
		.EXTRN	DBG\$GL_GET_LEX, DBG\$GL_PARTBPTR		
		.EXTRN	DBG\$GL_REDOC_R1		
		.EXTRN	DBG\$GB_DEF_OUT, DBG\$GW_DFLTLENG		
		.EXTRN	RST\$START_ADDR, DBG\$GL_ASCI_LEN		
		.EXTRN	DBG\$GB_LOC_TYPE		
		.EXTRN	DBG\$GL_CSP_PTR, DBG\$GL_LAST_LOC		
		.EXTRN	DBG\$GL_LAST_VAL		
		.WEAK	DBG\$GL_3B_SYSTEM		
		.PSECT	DBG\$CODE, NOWRT, SHR, PIC,0		
			07FC 00000		
		5E	04 04 C2 00002		
		51	AC DD 00005	SUBL2	#4, SP
		57	A1 9E 00009	MOVAB	PATHNAME, R1
		50	A1 9A 0000D	MOVZBL	B(R1), NAME_VECTOR
		57	6740 DE 00011	MOVAL	1(R1), R0
		56	61 9A 00015	MOVZBL	(NAME_VECTOR)[R0], NAME_VECTOR
		56	50 C2 00018	SUBL2	(R1), NAME_COUNT
		61	50 90 0001B	MOVB	R0, NAME_COUNT
			4002 8F BB 0001E	PUSHR	R0, (R1)
			00 02 FB 00022	CALLS	#^M<R1,SP>
			5A 6E DD 00029	MOVL	#2, DBG\$NPATHDESC_TO_CS
			52 6A 9A 0002C	MOVZBL	TOP_NAME, R10
			50 56 DD 0002F	MOVL	(R10), LENGTH
			0E 11 00032	BRB	NAME_COUNT, INDEX
			59 6740 DD 00034	1\$:	2\$
			51 69 9A 00038	MOVL	(NAME_VECTOR)[INDEX]. SUB_NAME
			52 05 15 0003B	MOVZBL	(SUB_NAME), R1
			EF 04 A142 9E 0003D	BLEQ	2\$
			52 50 F4 00042	MOVAB	4(R1)[LENGTH], LENGTH
			52 04 C7 00045	SOBGEQ	INDEX, 1\$
			01 A0 9F 00049	DIVL3	#4, LENGTH, R0
			00 01 FB 0004C	PUSHAB	1(R0)
			58 50 DD 00053	CALLS	#1, DBG\$GET_TEMPMEM
			08 BC 58 DD 00056	MOVL	R0_POINTER
					POINTER, @NAME_STRING

	88	52	90 0005A	MOV8	LENGTH, (POINTER)+	: 0384
		1D	11 0005D	BRB	4S	: 0385
	59	6746	D0 0005F	3\$: MOVL	(NAME_VECTOR)[INDEX], SUB_NAME	: 0387
			69 95 00063	TSTB	(SUB_NAME)	: 0388
			15 13 00065	BEQL	4S	: 0391
68	01	50	69 9A 00067	MOVZBL	(SUB_NAME), R0	: 0392
		A9	50 28 0006A	MOVC3	R0, T(SUB NAME), (POINTER)	: 0393
		50	69 9A 0006F	MOVZBL	(SUB NAME), R0	: 0385
		58	50 C0 00072	ADDL2	R0, POINTER	: 0397
		88 00000000'	EF D0 00075	MOVL	P.AAA, (POINTER)+	: 0398
		E0	56 F4 0007C	4\$: S0BGEQ	INDEX, 3S	
		50	6A 9A 0007F	MOVZBL	(R10), R0	
68	01	AA	50 28 00082	MOVC3	R0, 1(R10), (POINTER)	
			04 00087	RET		

; Routine Size: 136 bytes.    Routine Base: DBG\$CODE + 0000

```
268 0399 1 GLOBAL ROUTINE DBG$SEND_OF_CMD : NOVALUE =
269 0400 1 ++
270 0401 1 FUNCTIONAL DESCRIPTION:
271 0402 1 Resets all DEBUG context that is exclusive to a single
272 0403 1 DEBUG command. This includes resetting default
273 0404 1 modes from single line overrides back to the actual default
274 0405 1 modes and resetting a large number of context bits.
275 0406 1
276 0407 1 This routine also releases all temporary memory allocated in the
277 0408 1 course of processing the command, and it releases all unreferenced
278 0409 1 RST entries on the Temporary RST Entry List.
279 0410 1
280 0411 1 FORMAL PARAMETERS:
281 0412 1     none
282 0413 1
283 0414 1 IMPLICIT INPUTS:
284 0415 1     none
285 0416 1
286 0417 1 IMPLICIT OUTPUTS:
287 0418 1     The default modes, step-modes, and context bits are established.
288 0419 1     Some global storage is re-initialized and all excess storage is released.
289 0420 1
290 0421 1 ROUTINE VALUE:
291 0422 1     novalue
292 0423 1
293 0424 1 SIDE EFFECTS:
294 0425 1     none
295 0426 1
296 0427 1
297 0428 2 BEGIN
298 0429 2
299 0430 2
300 0431 2
301 0432 2
302 0433 2
303 0434 2
304 0435 2
305 0436 2
306 0437 2
307 0438 2
308 0439 2
309 0440 2
310 0441 2
311 0442 2
312 0443 2
313 0444 2
314 0445 2
315 0446 2
316 0447 2
317 0448 2
318 0449 2
319 0450 2
320 0451 2
321 0452 2
322 0453 2
323 0454 2
324 0455 2

    |+ Set the exit flag to true so that if an error occurs during
    | the processing of this routine, that error is perceived
    | as fatal. This routine guarantees the internal consistency
    | of DEBUG, and must succeed or give up.
    |
    |dbg$gv_control[dbg$v_control_exit] = TRUE;
    |
    | Clear the ALLOCATE flag. This is set during SET MODULE/ALLOCATE
    | to allow the allocation of additional memory.
    |
    |dbg$gv_control[dbg$v_control_allocate] = FALSE;
    |
    | Reset the Print control for DBGSPRINT. And flush out the print
    | buffer.
    |
    |DBGS$PRINT_CONTROL(DBGSK_PRT_RESET);
    |DBGS$FLUSHBUF();
    |
    | Reset mode level to user default level
    |
    |dbg$init_modes (override_mode, user_def_mode);
    |dbg$set_mod_lvl (user_def_mode);
    |
    | Reset search settings back to user default level
```

```
325      0456    2      dbg$init_search (override_search, user_def_search);
326      0457    2      dbg$set_search_lvl (user_def_search);
327      0458    2
328      0459    2      ; Reset define settings back to user default level
329      0460    2
330      0461    2      dbg$init_define (override_define, user_def_define);
331      0462    2      dbg$set_define_lvl (user_def_define);
332      0463    2
333      0464    2      dbg$set_context ();
334      0465    2
335      0466    2      DBGSREL_TEMPMEM();
336      0467    2      DBGSRST_TEMP RELEASE();
337      0468    2      dbg$gl_list [0] = 0;           ! Zero out the locations that hold breakpoint setting data.
338      0469    2      dbg$gl_list [1] = 0;
339      0470    2      dbg$gl_list [2] = 0;
340      0471    2      dbg$gl_lis_ptr = 0;          ! Zero current ptr to command arg list
341      0472    2      dbg$gl_asci_len = .dbg$gb_mod_ptr[mode_length]; ! Initialize ascii length
342      0473    2      dbg$gl_loctyp = -1;          ! Zero command override type.
343      0474    2      dbg$gw_locngth= 0;          ! And its associated length.
344      0475    2
345      0476    2      dbg$gl_module = 0;          ! Zero out global used to hold
346      0477    2
347      0478    2
348      0479    2      dbg$gl_modrstptr2 = 0;        ! module pointer during
349      0480    2
350      0481    2
351      0482    2
352      0483    2
353      0484    2
354      0485    2      dbg$gl_set_source = 0;        ! Clear the current primary cause there isn't one anymore
355      0486    2      dbg$gl_set_source2 = 0;       This flag is TRUE during a SET MODULE
356      0487    2      dbg$gl_current_primary = 0;   command.
357      0488    2      dbg$gb_set_module_flag = FALSE; Zero storage to hold array dimensions.
358      0489    2      zerocon (dbg$gl_dimenlst, 10); Zero storage to hold array dimensions
359      0490    2      zerocon (dbg$gl_nest_stack, 25); during nested subscript evaluation
360      0491    2      dbg$gl_nest_level = 0;        Nesting level of subscript expressions
361      0492    2      dbg$gb_set_break_flag = FALSE; set back to zero.
362      0493    2
363      0494    2
364      0495    2
365      0496    2
366      0497    2
367      0498    2
368      0499    2
369      0500    2
370      0501    2      dbg$sta_setcontext (0);      Initialize a flag saying whether we
371      0502    2
372      0503    2
373      0504    2      dbg$gv_control[dbg$v_control_exit] = FALSE; are in the middle of processing a
374      0505    2
END:
```

; Routine Size: 240 bytes. Routine Base: DBGSCODE + 0088

```

376 0506 1 GLOBAL ROUTINE dbg$end_of_line : NOVALUE =
377 0507 1 ++
378 0508 1 FUNCTIONAL DESCRIPTION:
379 0509 1 Calls dbg$end_of cmd to reset all single command context.
380 0510 1 Then frees the storage that was allocated to hold the command
381 0511 1 line. The top link of the command input stream is removed, but
382 0512 1 only if it is of type "buffer".
383 0513 1
384 0514 1 FORMAL PARAMETERS:
385 0515 1      none
386 0516 1
387 0517 1 IMPLICIT INPUTS:
388 0518 1      The head of the command argument list.
389 0519 1
390 0520 1 IMPLICIT OUTPUTS:
391 0521 1      none
392 0522 1
393 0523 1 ROUTINE VALUE:
394 0524 1      none
395 0525 1
396 0526 1 SIDE EFFECTS:
397 0527 1      Defaults are reestablished. Storage for input line is freed.
398 0528 1      A link may be removed from the command argument list.
399 0529 1 ---

400 0530 2 BEGIN
401 0531 2 LOCAL
402 0532 2      type;
403 0533 2
404 0534 2      dbg$end_of_cmd ();                                                  ! Perform end of command cleanup
405 0535 2      dbg$gv_control[dbg$gv_control_exit] = TRUE;                 ! Set the exit flag to return to CLI on errors
406 0536 2
407 0537 2      ! We only want to remove the top link of the cis if that link is a
408 0538 2      ! buffer of some flavor. If the top link is of type cis_rab, it has
409 0539 2      ! just been put there by an @... command and not yet read from.
410 0540 2
411 0541 2      type = .dbg$gl_cishead[cis$b_input_type];
412 0542 2      IF .type EQL cis_inpbuf
413 0543 2      OR .type EQL cis_acbuf
414 0544 2      OR .type EQL cis_while
415 0545 2      OR .type EQL cis_repeat
416 0546 2      OR .type EQL cis_if
417 0547 2      THEN
418 0548 2          dbg$cis_remove();
419 0549 2
420 0550 2      dbg$gv_control[dbg$gv_control_exit] = FALSE;                 ! Reset exit flag.
421 0551 2
422 0552 1 END;

```

		0004 00000	.ENTRY	DBGSEND OF LINE, Save R2	0506
FF02	52 0000000G	00 9E 00002	MOVAB	DBG\$GV CONTROL, R2	0534
	CF	00 FB 00009	CALLS	#0 DBGSEND_OF_CMD	0535
	62	10 88 0000E	BISB2	#16 DBG\$GV-CONTROL	0541
	50 0000000G	00 D0 00011	MOVL	DBG\$GL_CISHEAD, R0	

50	02	A0	9A 00018	MOVZBL	2(R0), TYPE
02		50	D1 0001C	CMPL	TYPE, #2
		14	13 0001F	BEQL	1\$
03		50	D1 00021	CMPL	TYPE, #3
		0F	13 00024	BEQL	1\$
05		50	D1 00026	CMPL	TYPE, #5
		0A	13 00029	BEQL	1\$
04		50	D1 0002B	CMPL	TYPE, #4
		05	13 0002E	BEQL	1\$
06		50	D1 00030	CMPL	TYPE, #6
		05	12 00033	BNEQ	2\$
0000V	CF	00	FB 00035 1\$:	CALLS	#0, DBG\$CIS_REMOVE
	62	10	8A 0003A 2\$:	BICB2	#16, DBG\$GV_CONTROL
			04 0003D	RET	

: Routine Size: 62 bytes.    Routine Base: DBG\$CODE + 0178

: 423            0553 1

```
425      0554 1 GLOBAL ROUTINE dbg$write_mem (dest_address, src_address, length) =  
426      0555 1 ++  
427      0556 1 FUNCTIONAL DESCRIPTION:  
428      0557 1 Writes a sequence of values (bytes) to memory in  
429      0558 1 the user program. The destination, source, and  
430      0559 1 number of bytes to write are all passed as parameters.  
431      0560 1  
432      0561 1 THE PROTECTION OF THE FIRST BYTE TO BE WRITTEN AND THE LAST  
433      0562 1 BYTE TO BE WRITTEN ARE BOTH CHECKED. THE STATUS OF BOTH PAGES  
434      0563 1 (THEY MAY BE THE SAME PAGE) IS SAVED. THEN THE VALUE IS WRITTEN  
435      0564 1 TO THE ADDRESS (THE PAGE PROTECTION IS CHANGED DURING THE  
436      0565 1 CHECKING OPERATION).  
437      0566 1  
438      0567 1 THEN, IF THE PROTECTION WAS CHANGED IN EITHER CASE, THE  
439      0568 1 PROTECTION IS REESTABLISHED. IF EVERYTHING WAS SUCCESSFUL,  
440      0569 1 THE ROUTINE RETURNS TRUE. OTHERWISE, IT RETURNS FALSE.  
441      0570 1  
442      0571 1 Formal Parameters:  
443      0572 1 dest_address - THE ADDRESS OF THE LOCATION TO BE CHANGED  
444      0573 1 src_address - The address of where the bytes are stored.  
445      0574 1 length - The number of bytes to be written.  
446      0575 1  
447      0576 1 IMPLICIT INPUTS:  
448      0577 1 None.  
449      0578 1  
450      0579 1 IMPLICIT OUTPUTS:  
451      0580 1 THE PAGE PROTECTION MAY BE MOMENTARILY ALTERED, THEN REINSTALLED.  
452      0581 1  
453      0582 1  
454      0583 1  
455      0584 1  
456      0585 1  
457      0586 1  
458      0587 1 --  
459      0588 1  
460      0589 1  
461      0590 1  
462      0591 1  
463      0592 1  
464      0593 1  
465      0594 1  
466      0595 1  
467      0596 1  
468      0597 1  
469      0598 1  
470      0599 1  
471      0600 1  
472      0601 1  
473      0602 1  
474      0603 1  
475      0604 1  
476      0605 1  
477      0606 1  
478      0607 1  
479      0608 1  
480      0609 1  
481      0610 1  
        dest_address : REF VECTOR[BYTE];  
        src_address : REF VECTOR[BYTE];  
        LOCAL  
        prot_status_1,  
        protection_1: BYTE,  
        prot_status_2,  
        protection_2: BYTE;  
        IF ((prot_status_1 = dbg$check_prot (.dest_address, protection_1)) NEQ 0)  
        AND ((prot_status_2 = dbg$check_prot (.dest_address + length - 1, protection_2)) NEQ 0)  
        THEN BEGIN  
        ++  
        ! PROTECTION HAS EITHER BEEN ALTERED SUCCESSFULLY, OR IT  
        ! DID NOT NEED TO BE ALTERED. NOW WRITE THE VALUE INTO THE  
        ! ADDRESS.  
        --
```

```

482      0611    CHSMOVE (.length, src_address [0], dest_address [0]);
483      0612
484      0613
485      0614    !++
486      0615    IF EITHER OF THE PROTECTION STATUSES SAY RESET THE PROTECTION
487      0616    (TO READ ONLY), THEN RESET THE PROTECTION ON THAT PAGE.
488      0617
489      0618
490      0619    --
491      0620    IF .prot_status_1 EQL dbg$k_reset_prot
492      0621    THEN BEGIN
493      0622        dbg$redo_prot (.dest_address, protection_1);
494      0623    END;
495      0624    IF .prot_status_2 EQL dbg$k_reset_prot
496      0625    THEN BEGIN
497      0626        dbg$redo_prot (.dest_address + .length - 1, protection_2);
498      0627    END;
499      0628
500      0629    RETURN TRUE
501      0630    ! User program updated correctly
502      0631    END
503      0632    RETURN FALSE
504      0633
505      0634
506      0635    ELSE
507      0636    END;

```

			07FC 00000	.ENTRY		:	0554
			5A 00000000G	MOVAB	DBG\$WRITE_MEM, Save R2,R3,R4,R5,R6,R7,R8,-		
			59 00000000G	MOVAB	R9,R10		
			5E	SUBL2	DBG\$CHECK PROT, R10		
		56	04	PUSHL	DBG\$REDO_PROT, R9		
		56	04	MOVL	#8, SP		
		6A	02	PUSHL	SP		
		58	02	CALLS	DEST_ADDRESS, R6		
		58	02	MOVL	R6		
		5C	13	BEQL	#2, DBG\$CHECK PROT		
		5C	13	PUSHAB	R0, PROT_STATUS_1		
		50	13	ADDL3	38		
		50	13	PUSHAB	PROTECTION_2		
		50	13	MOVL	LENGTH, R6, R0		
		50	13	BEQL	-1(R0)		
		56	04	PUSHAB	#2, DBG\$CHECK PROT		
		56	04	CALLS	R0, PROT_STATUS_2		
		56	04	MOVL	38		
		56	04	BEQL	LENGTH, @SRC_ADDRESS, (R6)		
		56	04	MOVC3	PROT_STATUS_T, #2		
		56	04	CMPL	18		
		56	04	BNEQ	#^M<R6,SP>		
		56	04	PUSHR	#2, DBG\$REDO PROT		
		56	04	CALLS	PROT_STATUS_2, #2		
		56	04	CMPL	28		
		56	04	BNEQ	PROTECTION_2		
		56	04	PUSHAB	LENGTH, R6, R0		
		56	04	ADDL3	-1(R0)		
		56	04	PUSHAB			

69	02	FB	00058		CALLS	#2, DBGSREDO_PROT
50	01	00	0005B	2\$:	MOVL	#1, R0
	04	0005E			RET	
	50	D4	0005F	3\$:	CLRL	R0
	04	00061			RET	

: 0633

: 0635

; Routine Size: 98 bytes,    Routine Base: DBGS CODE + 01B6

```
508      0636 1 GLOBAL ROUTINE dbg$set_context : NOVALUE =
509      0637 1
510      0638 1 ++ FUNCTIONAL DESCRIPTION:
511      0639 1 initializes context bits that are necessary for command
512      0640 1 processing. These bits are valid only during the processing
513      0641 1 of a single command. They are all reset after each command.
514      0642 1
515      0643 1
516      0644 1 CALLING SEQUENCE:
517      0645 1     dbg$set_context ()
518      0646 1
519      0647 1 INPUTS:
520      0648 1     none
521      0649 1
522      0650 1 IMPLICIT INPUTS:
523      0651 1     the names of the context bits that are to be turned off
524      0652 1
525      0653 1 OUTPUTS:
526      0654 1     none
527      0655 1
528      0656 1 IMPLICIT OUTPUTS:
529      0657 1     none
530      0658 1
531      0659 1 ROUTINE VALUE:
532      0660 1     novalue
533      0661 1
534      0662 1 SIDE EFFECTS:
535      0663 1     the context bits are set to false
536      0664 1
537      0665 1
538      0666 2
539      0667 2 BEGIN
540      0668 2     dbg$gl_context [dbg$k_all] = FALSE;
541      0669 2     dbg$gl_context [dbg$k_all_break] = FALSE;
542      0670 2     dbg$gl_context [dbg$k_all_trace] = FALSE;
543      0671 2     dbg$gl_context [dbg$k_all_watch] = FALSE;
544      0672 2     dbg$gl_context [dbg$k_break] = FALSE;
545      0673 2     dbg$gl_context [dbg$k_cancel] = FALSE;
546      0674 2     dbg$gl_context [dbg$k_examine] = FALSE;
547      0675 2     dbg$gl_context [dbg$k_language] = FALSE;
548      0676 2     dbg$gl_context [dbg$k_mode] = FALSE;
549      0677 2     dbg$gl_context [dbg$k_module] = FALSE;
550      0678 2     dbg$gl_context [dbg$k_override] = FALSE;
551      0679 2     dbg$gl_context [dbg$k_resignal] = FALSE;
552      0680 2     dbg$gl_context [dbg$k_scope] = FALSE;
553      0681 2     dbg$gl_context [dbg$k_search] = FALSE;
554      0682 2     dbg$gl_context [dbg$k_set_break] = FALSE;
555      0683 2     dbg$gl_context [dbg$k_step] = FALSE;
556      0684 2     dbg$gl_context [dbg$k_trce_call] = FALSE;
557      0685 2     dbg$gl_context [dbg$k_trace] = FALSE;
558      0686 2     dbg$gl_context [dbg$k_traceback] = FALSE;
559      0687 2     dbg$gl_context [dbg$k_watch] = FALSE;
560      0688 2     dbg$gl_context [dbg$k_trce_brch] = FALSE;
561      0689 2     dbg$gl_context [dbg$k_thread] = FALSE;
562      0690 2     dbg$gl_context [dbg$k_output] = FALSE;
563      0691 2     dbg$gl_context [dbg$k_log] = FALSE;
564      0692 2     dbg$gl_context [dbg$k_source] = FALSE;
565      0693 2     dbg$gl_context [dbg$k_margins] = FALSE;
```

DBGLEVEL1  
V04-000

{  
16-Sep-1984 01:27:02      VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:17:02      [DEBUG.SRC]DBGLEVEL1.B32;1

Page 18  
(6)

: 565      0693 2      dbg\$gl\_context [dbg\$k\_maxfiles] = FALSE;  
: 566      0694 1      END;

00000000G 00            1B            00            0000 00000      .ENTRY    DBGSSET\_CONTEXT, Save nothing  
                          00 F0 00002      INSV      #0, #0, #27, DBG\$GL\_CONTEXT  
                          04 0000B      RET

: Routine Size: 12 bytes,    Routine Base:    DBGS\$CODE + 0218

: 0636  
: 0693  
: 0694

```
568      0695 1 GLOBAL ROUTINE DBGSINIT_DEBUG: NOVALUE =
569      0696 1
570      0697 1 FUNCTION
571      0698 1   This routine drives the DEBUG initialization when DEBUG first comes up.
572      0699 1
573      0700 1 INPUTS
574      0701 1   NONE
575      0702 1
576      0703 1 OUTPUTS
577      0704 1   NONE
578      0705 1
579      0706 1
580      0707 2 BEGIN
581      0708 2
582      0709 2 BIND
583      0710 2   DBG_INPUT_DEVICE      = UPLIT BYTE ('ASCII' 'DBGSINPUT'),
584      0711 2   DBG_INP_DEV_SIZE    = %CHARCOUNT ('ASCII' 'DBGSINPUT')
585      0712 2   DBG_OUTPUT_DEVICE   = UPLIT BYTE ('ASCII' 'DBGSOUTPUT'),
586      0713 2   DBG_OUT_DEV_SIZE    = %CHARCOUNT ('ASCII' 'DBGSOUTPUT'),
587      0714 2
588      0715 2   SYS_INPUT_DEVICE     = UPLIT BYTE ('ASCII' 'SYSSINPUT'),
589      0716 2   SYS_INP_DEV_SIZE    = %CHARCOUNT ('ASCII' 'SYSSINPUT')
590      0717 2   SYS_OUTPUT_DEVICE   = UPLIT BYTE ('ASCII' 'SYSSOUTPUT'),
591      0718 2   SYS_OUT_DEV_SIZE    = %CHARCOUNT ('ASCII' 'SYSSOUTPUT'),
592      0719 2
593      0720 2 LOCAL
594      0721 2   DEF RADIX,
595      0722 2   DEVCHAR: REF BLOCK[,BYTE],          | Default radix
596      0723 2   DUMMY: VECTOR[2],                  | Device characteristics field
597      0724 2   DUMMY_BUFFER: VECTOR[256,BYTE], | Output area for STRNLOG
598      0725 2   EVNT FLAG,
599      0726 2   FILESPEC: DBGSSTG_DESC,           | String descriptor
600      0727 2   HEADER: REF DEFINESHADER,        | Header block for define
601      0728 2   ITEM: BLOCK[6, LONG],            | symbol table.
602      0729 2   LEN,                         | Item list for SGETSYI
603      0730 2   OPEN_STATUS,
604      0731 2   OPEN_STATUS,                   | Save the failing status from SOPEN
605      0732 2   OUTPUT_STATUS,                 | DBG$INPUT
606      0733 2   OUTPUT_STATUS,                 | Save the failing status from SCREATE
607      0734 2   | DBG$OUTPUT
608      0735 2   SDBGINIT_STGDESC: BLOCK[8,BYTE], | String descriptor
609      0736 2   SDBGINIT_STG: VECTOR [9, BYTE], | String in string descriptor
610      0737 2   STATUS,
611      0738 2   STATUS|,
612      0739 2   VERSION_BUFFER: VECTOR[8, BYTE];
613      0740 2
614      0741 2
615      0742 2
616      0743 2   ! Initialize an area of free storage. This must be done first since many
617      0744 2   ! of the things below will call the memory allocation routines.
618      0745 2
619      0746 2   DBGSINIT_MEMORY();
620      0747 2
621      0748 2
622      0749 2   ! Initialize the bit that says whether we are on a V4 system.
623      0750 2   ! We call the system service SGETSYI to find out this information.
624      0751 2
```

```
625      0752 2 | Note - the code to call GETSYI is commented out because this
626      0753 2 | turned out to be unreliable (could get back a variety of things,
627      0754 2 | such as "V3.5", "X3.5", "X29T", "X4.0", "V4.0"). We are instead
628      0755 2 | just using a link-time symbol (see below).
629      0756 2
630      0757 2
631      0758 2
632      0759 2
633      0760 2
634      0761 2
635      0762 2
636      0763 2
637      0764 2
638      0765 2
639      0766 2
640      0767 2
641      0768 2
642      0769 2
643      0770 2
644      0771 2
645      0772 2
646      0773 2
647      0774 2
648      0775 2
649      0776 2
650      0777 2
651      0778 2
652      0779 2
653      0780 2
654      0781 2
655      0782 2
656      0783 2
657      0784 2
658      0785 2
659      0786 2
660      0787 2
661      0788 2
662      0789 2
663      0790 2
664      0791 2
665      0792 2
666      0793 2
667      0794 2
668      0795 2
669      0796 2
670      0797 2
671      0798 2
672      0799 2
673      0800 2
674      0801 2
675      0802 2
676      0803 2
677      0804 2
678      0805 2
679      0806 2
680      0807 2
681      0808 3

| Note - the code to call GETSYI is commented out because this
| turned out to be unreliable (could get back a variety of things,
| such as "V3.5", "X3.5", "X29T", "X4.0", "V4.0"). We are instead
| just using a link-time symbol (see below).
| ITEM[0,0,16,0] = 8;
| ITEM[0,16,16,0] = $YIS VERSION;
| ITEM[1,0,32,0] = VERSION_BUFFER;
| ITEM[2,0,32,0] = LEN;
| CHSFILL(0, 12, ITEM[3, A]);
| STATUS = LIBSGET EF(EVNT-FLAG);
| IF NOT .STATUS THEN EVNT-FLAG = 0;
| STATUS = $GETSYI(EFN=.EVNT_FLAG, ITMLST=ITEM);
| IF .STATUS
| THEN
| Version 3 systems will return "V3.x" in VERSION_BUFFER.
| ! DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4] = NOT
| ! ((.VERSION_BUFFER[0] EQ 'V') AND (.VERSION_BUFFER[1] EQ '3'))
| ! ELSE
| ! $GETSYI failed. Make a guess that we are a 3B system.
| ! DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4] = 1;
| ! LIBSFREE_EF(EVNT_FLAG);
|
| Initialize the bit that says whether we are on a 3B system.
| We rely on a link-time symbol DBG$GL_3B_SYSTEM.
DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4] = DBG$GL_3B_SYSTEM;
|
| Initialize the global which says whether we are trying to do
| keypad input.
DBG$GB_KEYPAD_INPUT = .DBG$GV_CONTROL[DBG$V_CONTROL_VERSION_4];
|
| Open the input device for reading. If the OPENS and CONNECTS cannot be
| done successfully for logical devices 'DBG$INPUT' and 'DBG$OUTPUT', then
| try 'SYSSINPUT' and 'SYSSOUTPUT'. If these fail, signal an error. This
| causes a return to the command line interpreter in the operating system.
DBG$GL_INPFAB[FABSL_FNA] = DBG_INPUT_DEVICE;
DBG$GL_INPFAB[FABSB_FNS] = DBG_INP_DEV_SIZE;
OPEN STATUS = SOPEN TFAB = DBG$GL_INPFAB;
IF NOT .OPEN_STATUS
THEN
BEGIN
DBG$GL_INPFAB[FABSL_FNA] = SYS_INPUT_DEVICE;
DBG$GL_INPFAB[FABSB_FNS] = SYS_INP_DEV_SIZE;
STATUS = SOPEN (FAB = DBG$GL_INPFAB);
IF NOT .STATUS THEN SEXIT(CODE = .STATUS OR FATAL_BIT);
```

```
682      0809      END;  
683      0810        
684      0811        
685      0812      ! Connect the input file.  
686      0813      !  
687      0814      DBG$GL_INPRAB[RABSL_FAB] = DBG$GL_INPFAB;  
688      0815      STATUS = $CONNECT(RAB = DBG$GL_INPRAB);  
689      0816      IF NOT .STATUS THEN SEXIT(CODE = .STATUS OR FATAL_BIT);  
690      0817        
691      0818        
692      0819      ! CREATE and OPEN the output file.  
693      0820      !  
694      0821      DBG$GL_OUTPFAB [FABSL_FNA] = DBG_OUTPUT DEVICE;  
695      0822      DBG$GL_OUTPFAB [FABS8_FNS] = DBG_OUT DEV SIZE;  
696      0823      OUTPUT_STATUS = SCREATE (FAB = DBG$GL_OUTPFAB);  
697      0824      IF NOT .OUTPUT_STATUS  
698      0825      THEN  
699      0826      BEGIN  
700      0827      DBG$GL_OUTPFAB [FABSL_FNA] = SYS_OUTPUT DEVICE;  
701      0828      DBG$GL_OUTPFAB [FABS8_FNS] = SYS_OUT DEV_SIZE;  
702      0829      STATUS = SCREATE (FAB = DBG$GL_OUTPFAB);  
703      0830      IF NOT .STATUS THEN SEXIT(CODE = .STATUS OR FATAL_BIT);  
704      0831      END;  
705      0832        
706      0833        
707      0834      ! CONNECT the output file.  
708      0835      !  
709      0836      DBG$GL_OUTPRAB[RABSL_FAB] = DBG$GL_OUTPFAB;  
710      0837      STATUS = $CONNECT(RAB = DBG$GL_OUTPRAB);  
711      0838      IF NOT .STATUS THEN SEXIT(CODE = .STATUS OR FATAL_BIT);  
712      0839        
713      0840        
714      0841      ! We need to delay this message output till SYSSINPUT, SYSSOUTPUT are  
715      0842      established. Otherwise, DBGSPUTMSG in DBGSFINAL_HANDL does not know  
716      0843      where to output the message.  
717      0844        
718      0845      IF NOT .OPEN_STATUS THEN SIGNAL(DBGS_UNAOPEDBG1, 0, .OPEN_STATUS);  
719      0846      IF NOT .OUTPUT_STATUS THEN SIGNAL(DBGS_UNACREDBG0, 0, .OUTPUT_STATUS);  
720      0847        
721      0848        
722      0849      ! Get the terminal width.  
723      0850      !  
724      0851      DEVCHAR = DBG$GL_OUTPFAB[FABSL_DEV];  
725      0852      IF .DEVCHAR[DEV$V_TRM]  
726      0853      THEN  
727      0854      BEGIN  
728      0855      LOCAL  
729      0856      DEV_DESC: VECTOR[2, LONG],  
730      0857      INFO_4: VECTOR[4, LONG],  
731      0858      RETURN_LENGTH;  
732      0859        
733      0860      DEV_DESC[0] = XX'010E0000' OR DBG$GL_OUTPFAB [FABS8_FNS];  
734      0861      DEV_DESC[1] = DBG$GL_OUTPFAB [FABSL_FNA];  
735      0862      INFO_4[0] = DVIS DEVBOFSIZ^16 OR 4;  
736      0863      INFO_4[1] = DBG$SRC_TERM WIDTH;  
737      0864      INFO_4[2] = RETURN_LENGTH;  
738      0865      INFO_4[3] = 0;
```

```
739      0866      STATUS = SGETDVI(DEVNAM=DEV_DESC, ITMLST=INFO_4);  
740      0867      IF NOT .STATUS THEN SIGNAL(.STATUS);  
741      0868      END  
742      0869  
743      0870  
744      0871      ELSE DBGSSRC_TERM_WIDTH = 80;  
745      0872  
746      0873  
747      0874      ! Set the flag that says resignal all exceptions from the user  
748      0875      program except for user-set breakpoints and tracepoints.  
749      0876      DBGSGB_RESIGNAL = TRUE;  
750      0877  
751      0878  
752      0879  
753      0880      ! Initialize the define settings.  
754      0881      DBG$SET_DEFINE_DEF();  
755      0882  
756      0883  
757      0884      ! Initialize the DEFINE symbol table by allocating space for  
758      0885      the header blocks, and initializing the fields to be zero.  
759      0886      ! Note that this must be done before DBG$SET_LANG.  
760      0887      !  
761      0888      HEADER = DBGSGET MEMORY (DBGSK_DEFINE_HEADER_SIZE_W);  
762      0889      HEADER [DEFSA_NEXT_LINK] = 0;  
763      0890      HEADER [DEFSA_PREV_LINK] = 0;  
764      0891      HEADER [DEFSA_DEFINE_LIST] = 0;  
765      0892      DBG$GL_GLOBAL_DEFINE_PTR = .HEADER;  
766      0893      HEADER = DBGSGET MEMORY (DBGSK_DEFINE_HEADER_SIZE_W);  
767      0894      HEADER [DEFSA_NEXT_LINK] = 0;  
768      0895      HEADER [DEFSA_PREV_LINK] = 0;  
769      0896      HEADER [DEFSA_DEFINE_LIST] = 0;  
770      0897      DBG$GL_LOCAL_DEFINE_PTR = .HEADER;  
771      0898  
772      0899  
773      0900      ! Set the default language, namely MACRO.  
774      0901      DBG$SET_LANG(0,DBGSK_MACRO);  
775      0902  
776      0903  
777      0904  
778      0905      ! Set all the single command context bits to FALSE. These bits refer to  
779      0906      context that is valid only during a single command, not across multiple  
780      0907      commands.  
781      0908      !  
782      0909      DBG$SET_CONTEXT();  
783      0910  
784      0911  
785      0912  
786      0913      ! Initialize the new eventpoint data structures.  
787      0914      DBGSEVENT_INITIALIZATION();  
788      0915  
789      0916  
790      0917  
791      0918      ! Initialize the Command Input Stream to DBG$INPUT  
792      0919      DBG$GL_CISHEAD = DBGSGET MEMORY ((CIS_ELEMENTS+3)/ZUPVAL);  
793      0920      DBG$GL_CISHEAD[CISSA_NEXT_LINK] = 0;  
794      0921      DBG$GL_CISHEAD[CISBB_INPUT_TYPE] = CIS_DBG$INPUT;  
795      0922 2
```

```

796      0923 2
797      0924 2
798      0925 2
799      0926 2
800      0927 2
801      0928 2
802      0929 2
803      0930 2
804      0931 2
805      0932 2
806      0933 2
807      0934 2
808      0935 2
809      0936 2
810      0937 2
811      0938 2
812      0939 2
813      0940 2
814      0941 2
815      0942 2
816      0943 2
817      0944 2
818      0945 2
819      0946 2
820      0947 2
821      0948 2
822      0949 2
823      0950 2
824      0951 2
825      0952 2
826      0953 2
827      0954 2
828      0955 2
829      0956 2
830      0957 2
831      0958 2
832      0959 4
833      0960 4
834      0961 4
835      0962 4
836      0963 4
837      0964 4
838      0965 4
839      0966 4
840      0967 4
841      0968 4
842      0969 4
843      0970 4
844      0971 4
845      0972 4
846      0973 4
847      0974 4
848      0975 4
849      0976 4
850      0977 4
851      0978 4
852      0979 4

    DBGSGL_CISHEAD[CISSA_INPUT_PTR] = DBGSGL_INPRAB;

    ! Initialize the OUTPUT configuration
    DBGSSET_OUT_DEF();

    ! Note - processing initialization files must be done last. If there was
    ! an initialization file, add it to the command input stream. For DEBUG,
    ! the initialization file is specified by the logical name DBGSINIT, and
    ! for SUPERDEBUG, it is specified by the logical name SDBGSINIT.

    DUMMY[0] = %X'010E0000'+256;
    DUMMY[1] = DUMMY_BUFFER;

    ! We need to allocate space for the file name and copy 'DBGSINIT' or
    ! 'SDBGSINIT' into this space. The reason for this is that DBGCIS_REMOVE
    ! will free up the space. Also, fill in the string descriptor to
    ! be used in SYS$TRNLOG. Note - do NOT replace this with a %ASCID
    ! declaration. %ASCID causes the code to be non-shareable and thus
    ! degrades performance.

    DBGSGL_IND_COM_FILE = DBGSGET_MEMORY(3);
    SDBGINIT_STGDESC [DSC$8_CLASS] = DSC$K_CLASS_S;
    SDBGINIT_STGDESC [DSC$8_DTYPE] = DSC$K_DTYPE_T;
    SDBGINIT_STGDESC [DSC$A_POINTER] = SDBGINIT_STG;
    IF .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
    THEN
        BEGIN
            SDBGINIT_STGDESC [DSC$W_LENGTH] = 9;
            CHSMOVE(9, UPLIT BYTE(%ASCII 'SDBGSINIT'), SDBGINIT_STG);
            STATUS = $YS$TRNLOG(SDBGINIT_STGDESC, 0, DUMMY, 0, 0, 0);
            IF .STATUS EQL SSS_NORMAL
            THEN
                BEGIN
                    DBGSGL_IND_COM_FILE[0] = 9;
                    CHSMOVE(9, UPLIT BYTE(%ASCII 'SDBGSINIT'), DBGSGL_IND_COM_FILE[1]);
                    DBGCIS_CONNECT(IF(FALSE));
                END;
        END
    ELSE
        BEGIN
            SDBGINIT_STGDESC [DSC$W_LENGTH] = 8;
            CHSMOVE(8, UPLIT BYTE(%ASCII 'DBGSINIT'), SDBGINIT_STG);
            STATUS = $YS$TRNLOG(SDBGINIT_STGDESC, 0, DUMMY, 0, 0, 0);
            IF .STATUS EQL SSS_NORMAL
            THEN
                BEGIN
                    DBGSGL_IND_COM_FILE[0] = 8;
                    CHSMOVE(8, UPLIT BYTE(%ASCII 'DBGSINIT'), DBGSGL_IND_COM_FILE[1]);
                    DBGCIS_CONNECT(IF(FALSE));
                END;
        END;
    END;

```

```
853      0980 2
854      0981
855      0982
856      0983
857      0984
858      0985
859      0986
860      0987
861      0988
862      0989
863      0990
864      0991
865      0992
866      0993
867      0994 1

END:

; Initialization is complete and successful. Output the DEBUG header
; message with the version number and return.

IF .DBGSGV_CONTROL[DBGSV_CONTROL_SDBG]
THEN
    $FAO_TT_OUT('!/ VAX SUPERDEBUG Version 4.0-8!/')
ELSE
    $FAO_TT_OUT('!/ VAX DEBUG Version 4.0-8!/');
RETURN;
END;
```

```

        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

54 54 55 55 50 50 4E 4F 24 47 42 44 00004 P.AAB: .ASCII \DBG$INPUT\
54 54 55 50 54 55 4E 4F 24 47 42 44 0000D P.AAC: .ASCII \DBG$OUTPUT\
54 54 49 4E 49 24 47 42 44 53 59 53 00017 P.AAD: .ASCII \SYS$INPUT\
54 54 49 4E 49 24 47 42 44 53 59 53 00020 P.AAE: .ASCII \SYS$OUTPUT\
54 54 49 4E 49 24 47 42 44 53 59 53 0002A P.AAF: .ASCII \SDBG$INIT\
54 54 49 4E 49 24 47 42 44 53 59 53 00033 P.AAG: .ASCII \SDBG$INIT\
54 54 49 4E 49 24 47 42 44 53 59 53 0003C P.AAH: .ASCII \DBG$INIT\
54 54 49 4E 49 24 47 42 44 53 59 53 00044 P.AAI: .ASCII \DBG$INIT\
54 54 49 4E 49 24 47 42 44 53 59 53 0004C P.AAJ: .BYTE 34
45 64 52 45 50 55 53 20 58 41 56 20 09 2F 21 0004D .ASCII \!/\<9>\ VAX SUPERDEBUG Version 4.0-8!/\:
30 2E 34 20 6E 6F 69 73 72 65 56 20 47 55 42 0005C
          2F 21 38 2D 0006B
          1D 0006F P.AAK: .BYTE 29
56 20 47 55 42 45 44 20 58 41 56 20 09 2F 21 00070 .ASCII \!/\<9>\ VAX DEBUG Version 4.0-8!/\:
2F 21 38 20 30 2E 34 20 6E 6F 69 73 72 65 0007F

        DBG_INPUT DEVICE= P.AAB
        DBG_INP DEV SIZE= 9
        DBG_OUTPUT DEVICE= P.AAC
        DBG_OUT DEV SIZE= 10
        SYS_INPUT DEVICE= P.AAD
        SYS_INP DEV SIZE= 9
        SYS_OUTPUT DEVICE= P.AAE
        SYS_OUT DEV SIZE= 10
        .EXTRN SYS$OPEN, SYS$EXIT
        .EXTRN SYS$CONNECT, SYS$CREATE
        .EXTRN SYS$GETDV1

        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

        OFFC 00000
        .ENTRY  DBG$INIT_DEBUG, Save R2,R3,R4,R5,R6,R7,R8,- : 0695
        SB 00000000G 00 9E 00002 MOVAB SY$SEXIT, R11
        SA 00000000G 00 9E 00009 MOVAB DBG$GV_CONTROL, R10
        S9 00000000G 00 9E 00010 MOVAB DBG$GL-INPFAB, R9
        S8 00000000G 00 9E 00017 MOVAB DBG$GL-OUTPFAB+44, R8
        S7 00000000G EF 9E 0001E MOVAB DBG_INPUT_DEVICE, R7

```

01	AA	01	AA	00000000G	5E	FE9C	CE	9E	00025		MOVAB	-356(SP), SP	
50				00	00	FB	0002A			CALLS	#0, DBGSINIT MEMORY	0746	
				04	04	FO	00031			INSV	#DBGSGL_3B SYSTEM, #4, #1, DBGSGV_CONTROL+1	0785	
				01	01	FF	0003B			EXTZV	#4, #1, DBGSGV CONTROL+1, R0	0791	
				00	50	90	00041			MOVB	R0, DBGSGB KEYPAD INPUT		
				2C	A9	67	9E	00048		MOVAB	DBG_INPUT DEVICE, DBGSGL_INPFAB+44	0799	
				34	A9	09	90	0004C		MOVB	#9, DBGSGL_INPFAB+52	0800	
				00000000G	00	59	DD	00050		PUSHL	R9	0801	
				53	53	50	DD	00052		CALLS	#1, SYSSOPEN		
				1F	A7	53	E8	0005C		MOVL	R0, OPEN STATUS		
				34	A9	09	90	00064	13	BLBS	OPEN STATUS 1\$	0802	
				00000000G	00	59	DD	00068		MOVAB	SYS_INPUT DEVICE, DBGSGL_INPFAB+44	0805	
				56	56	01	FB	0006A		MOVB	#9, DBGSGL_INPFAB+52	0806	
				07	56	50	DO	00071		PUSHL	R9	0807	
				7E	56	56	E8	00074		CALLS	#1, SYSSOPEN		
				6B	04	C9	00077		MOVL	R0, STATUS			
				00000000G	00	69	9E	0007E	18:	BLBS	STATUS, 1\$	0808	
				00000000G	00	9F	00085		BISL3	#4, STATUS, -(SP)			
				00000000G	00	01	FB	0008B		CALLS	#1, SYSSEXIT		
				56	56	50	DO	00092		MOVAB	DBG\$GL_INPFAB, DBG\$GL_INPRAB+60	0814	
				07	56	56	E8	00095		PUSHAB	DBG\$GL_INPRAB	0815	
				7E	56	04	C9	00098		CALLS	#1, SYSSCONNECT		
				6B	6B	01	FB	0009C		MOVL	R0, STATUS		
				08	68	A7	9E	0009F	28:	BLBS	STATUS, 2\$	0816	
				A8	09	0A	90	000A3		MOVAB	#4, STATUS, -(SP)		
					D4	A8	9F	000A7		PUSHAB	#1, SYSSEXIT		
				00000000G	00	01	FB	000AA		CALLS	DBG_OUTPUT DEVICE, DBG\$GL_OUTPFAB+44	0821	
				52	52	50	DO	000B1		MOVL	#10, DBGSGL_OUTPFAB+52	0822	
				1F	1F	52	E8	000B4		BLBS	DBG\$GL_OUTPFAB	0823	
				6B	6B	A7	9E	000B7		MOVAB	#1, SYSSCREATE		
				08	A8	0A	90	000BB		MOVB	R0, OUTPUT STATUS		
					D4	A8	9F	000BF		PUSHAB	OUTPUT STATUS, 3\$		
				00000000G	00	01	FB	000C2		CALLS	SYS_OUTPUT DEVICE, DBG\$GL_OUTPFAB+44	0824	
				56	56	50	DO	000C9		MOVL	#10, DBGSGL_OUTPFAB+52	0827	
				07	07	56	E8	000CC		BLBS	DBG\$GL_OUTPFAB	0828	
				7E	56	04	C9	000CF		MOVAB	#1, SYSSCREATE		
				6B	6B	01	FB	000D3		CALLS	R0, STATUS		
				00000000G	00	A8	9E	000D6	38:	MOVL	STATUS, 3\$	0830	
				00000000G	00	00	9F	000DE		BLBS	#4, STATUS, -(SP)		
				00000000G	00	01	FB	000E4		MOVAB	#1, SYSSEXIT		
				56	56	50	DO	000EB		PUSHAB	DBG\$GL_OUTPFAB, DBG\$GL_OUTPRAB+60	0836	
				07	07	56	E8	000EE		CALLS	#1, SYSSCONNECT	0837	
				7E	56	04	C9	000F1		MOVL	R0, STATUS		
				6B	6B	01	FB	000F5		BLBS	STATUS, 4\$	0838	
				11	11	53	E8	000F8	48:	BISL3	#4, STATUS, -(SP)		
				00000000G	00	53	DD	000FB		CALLS	#1, SYSSEXIT		
				00028FDB	8F	7E	D4	000FD		BLBS	OPEN_STATUS, 5\$		
				11	03	FB	00105		PUSHL	OPEN_STATUS			
				00000000G	00	52	E8	0010C	58:	CLRL	-(SP)		
				00028FE3	52	52	DD	0010F		PUSHL	#167899		
				11	7E	D4	00111		CALLS	#3, LIB\$SIGNAL			
				00000000G	00	03	FB	00113		BLBS	OUTPUT_STATUS, 6\$	0846	
				00028FE3	03	FB	00119		PUSHL	OUTPUT_STATUS			
										CLRL	-(SP)		
										PUSHL	#167907		
										CALLS	#3, LIB\$SIGNAL		

4C		50	14	A8	9E	00120	68:	MOVAB	DBGSGL_OUTPFAB+64, DEVCHAR	0851			
		60	08	02	E1	00124		BBC	#2, (DEVCHAR) 78	0852			
14	AE	50	010E0000	A8	9A	00128		MOVZBL	DBGSGL_OUTPFAB+52, RO	0860			
		18	AE	8F	C9	0012C		BISL3	#17694720, RO, DEV_DESC	0861			
		04	AE	00080004	68	D0	00135	MOVL	DBGSGL_OUTPFAB+44, DEV_DESC+4	0862			
		08	AE	BF	D0	00139		MOVAB	#524292, INFO_4	0863			
		OC	AE	00	9E	00141		MOVAB	DBGSSRC TERM WIDTH, INFO_4+4	0864			
				6E	9E	00149		MOVAB	RETURN LENGTH, INFO_4+8	0865			
			10	AE	D4	0014D		CLRL	INFO_4+12	0866			
				7E	7C	00150		CLRQ	-(SP)				
				7E	7C	00152		CLRQ	-(SP)				
			14	AE	9F	00154		PUSHAB	INFO_4				
			28	AE	9F	00157		PUSHAB	DEV_DESC				
		00000000G	00	7E	7C	0015A		CLRQ	-(SP)				
			56	08	FB	0015C		CALLS	#8, SYSSGETDVI				
			13	50	D0	00163		MOVL	RO, STATUS	0867			
		00000000G	00	56	E8	00166		BLBS	STATUS, 88				
				56	DD	00169		PUSHL	STATUS				
		00000000G	00	01	FB	0016B		CALLS	#1, LIBSSIGNAL				
			08	08	11	00172		BRB	88				
		00000000G	00	8F	9A	00174	78:	MOVZBL	#80, DBGSSRC TERM WIDTH	0852			
			00	01	90	0017C	88:	MOVB	#1, DBGSGB RESIGNAL	0871			
		00000000G	00	00	FB	00183		CALLS	#0, DBGSET_DEFINE_DEF	0877			
			03	03	DD	0018A		PUSHL	#3	0882			
		00000000G	00	01	FB	0018C		CALLS	#1, DBGSGET_MEMORY	0889			
			60	60	7C	00193		CLRQ	(HEADER)	0890			
		00000000G	00	08	A0	D4	00195	CLRL	8(HEADER)	0892			
			50	50	D0	00198		MOVL	HEADER, DBGSGL_GLOBAL_DEFINE_PTR	0893			
		00000000G	00	03	DD	0019F		PUSHL	#3	0894			
			01	01	FB	001A1		CALLS	#1, DBGSGET_MEMORY	0895			
		00000000G	00	60	7C	001A8		CLRQ	(HEADER)	0897			
			08	A0	D4	001AA		CLRL	8(HEADER)	0898			
		00000000G	00	50	D0	001AD		MOVL	HEADER, DBGSGL_LOCAL_DEFINE_PTR	0903			
			7E	7E	7C	001B4		CLRQ	-(SP)				
		0000V	CF	02	FB	001B6		CALLS	#2, DBGSET_LANG				
			FE34	CF	00	FB	001BB	CALLS	#0, DBGSET_CONTEXT				
		00000000G	00	00	FB	001C0		CALLS	#0, DBGSEVENT_INITIALIZATION	0915			
			OE	DD	001C7		PUSHL	#14	0920				
		00000000G	00	01	FB	001C9		CALLS	#1, DBGSGET_MEMORY				
			50	D0	001D0		MOVL	RO, DBGSGL_CISHEAD					
		00000000G	00	08	A0	D4	001D7	CLRL	8(R0)	0921			
			02	A0	94	001DA		CLRB	2(R0)	0922			
		04	A0	00000000G	00	9E	001DD	MOVAB	DBGSGL_INPRAB, 4(R0)	0923			
		00000000G	00	00	FB	001E5		CALLS	#0, DBGSET_OUT_DEF	0928			
		F8	AD	010E0100	8F	D0	001EC	MOVL	#17694976, DUMMY	0936			
		FC	AD	5C	AE	9E	001F4	MOVAB	DUMMY_BUFFER, DUMMY+4	0937			
				03	DD	001F9		PUSHL	#3	0947			
		00000000G	00	01	FB	001FB		CALLS	#1, DBGSGET_MEMORY				
		00000000G	00	50	D0	00202		MOVL	RO, DBGSGL_IND_COM_FILE				
		32	AE	010E	8F	B0	00209	MOVW	#270, SDBGINIT_STGDESC+2	0949			
		34	AE	24	AE	9E	0020F	MOVAB	SDBGINIT_STG, SDBGINIT_STGDESC+4	0950			
37				01	E1	00214		BBC	#1, DBGSIG CONTROL, 98	0951			
				09	B0	00218		MOVW	#9, SDBGINIT_STGDESC	0954			
				09	28	0021C		MOVCL3	#9, P.AAF, SDBGINIT_STG	0955			
				7E	7C	00222		CLRQ	-(SP)				
				7E	D4	00224		CLRL	-(SP)				
24	AE	30	AE	26	A7	9F	00226	PUSHAB	DUMMY	0956			

L 2  
16-Sep-1984 01:27:02 VAX-11 Bliss-32 v4.0-742  
14-Sep-1984 12:17:02 [DEBUG.SRC]DBGLEVEL1.B32;1

; Routine Size: 671 bytes, Routine Base: DBGSCODE + 0224

DBGLEVEL1  
V04-000

M 2

16-Sep-1984 01:27:02  
14-Sep-1984 12:17:02

VAX-11 Bliss-32 v4.0-742  
[DEBUG.SRC]DBGLEVEL1.B32;1

Page 28  
(8)

: 869 0995 1 GLOBAL ROUTINE dbg\$ins\_opcodes (user\_pc\_value) : NOVALUE =  
: 870 0996 2 BEGIN  
: 871 0997 2 0  
: 872 0998 1 END:

0000 00000  
04 00002

.ENTRY DBG\$INS\_OPCODES, Save nothing  
.RET

; Routine Size: 3 bytes, Routine Base: DBG\$CODE + 04C3

; 0995  
; 0998

```
874      0999 1 GLOBAL ROUTINE dbg$conv_r_50( OPWORD, DST_PTR ) =  
875      1000 1 ++  
876      1001 1 FUNCTIONAL DESCRIPTION:  
877      1002 1 THIS ROUTINE IS A SPECIAL-PURPOSE RAD50-TO-ASCII CONVERSION  
878      1003 1 ROUTINE. IT TAKES A LONGWORD CONTAINING 2 RAD50 WORDS,  
879      1004 1 CONVERTS THEM TO A 6-character STRING, AND 'STUFFS' THESE  
880      1005 1 characters INTO THE BYTE VECTOR WE ARE PASSED A POINTER TO.  
881      1006 1  
882      1007 1 WARNING:  
883      1008 1 THE REST OF THE CODE WHICH INTERFACES TO THIS USES THE  
884      1009 1 LITERAL 'OP_CH_SIZE' TO REFER TO THIS 6-character STRING,  
885      1010 1 BUT THIS SIZE IS IMPLICIT IN THE CODE HERE. IF THIS  
886      1011 1 CODE IS CHANGED, THE LITERAL MUST ALSO BE CHANGED. JUST  
887      1012 1 CHANGING THE LITERAL IS NOT ENOUGH.  
888      1013 1  
889      1014 1 CALLING SEQUENCE:  
890      1015 1     dbg$conv_r_50 ( LONGWORD, BYTE_POINTER );  
891      1016 1  
892      1017 1 INPUTS:  
893      1018 1     OPWORD - THE LONGWORD WHICH CONTAINS THE 6 RAD50 characters.  
894      1019 1     - THIS WORD COMES FROM THE OP_NAME FIELD OF THE  
895      1020 1     dbg$GB_OPINFO DATA VECTOR.  
896      1021 1     DST_PTR - ANY BYTE ADDRESS INTO WHICH THIS ROUTINE WILL  
897      1022 1     STUFF THE 6 ASCII characters OBTAINED FROM OPWORD.  
898      1023 1  
899      1024 1 IMPLICIT INPUTS:  
900      1025 1     OP_CH_SIZE - FROM VAXOPS.REQ, WHICH MUST BE 6 TO MATCH THIS CODE.  
901      1026 1  
902      1027 1 OUTPUTS:  
903      1028 1     NONE.  
904      1029 1  
905      1030 1 IMPLICIT OUTPUTS:  
906      1031 1     THE 6 ASCII chars ARE STUFFED BACK INTO THE USER-SUPPLIED  
907      1032 1     STRING.  
908      1033 1  
909      1034 1 ROUTINE VALUE:  
910      1035 1     The number of non-blank characters stuffed into  
911      1036 1     the output string.  
912      1037 1  
913      1038 1 SIDE EFFECTS:  
914      1039 1     SEE IMPLICIT OUTPUTS.  
915      1040 1 --  
916      1041 1 BEGIN  
917      1042 2  
918      1043 2  
919      1044 2  
920      1045 2 MAP  
921      1046 2     DST_PTR : REF VECTOR[.BYTE]; ! WHERE TO STUFF THE chars.  
922      1047 2  
923      1048 2  
924      1049 2 LOCAL  
925      1050 2     non_blanks  
926      1051 2     J, ! INDEX.  
927      1052 2     U, ! THE LONGWORD.  
928      1053 2     PIR : REF VECTOR[.BYTE];  
929      1054 2  
930      1055 2 BIND  
         DIVTAB = UPLIT(1, X0'50', X0'3100') : VECTOR;
```

```

931    1056 2      ! JUST EXTRACT EACH OF THE TWO WORDS, CONVERT THEM,
932    1057 2      ! AND STUFF BACK THE RESTULS.
933    1058 2
934    1059 2      PTR = .DST_PTR;
935    1060 2      non_blanks = 0;
936    1061 2
937    1062 2      INCR K FROM 0 TO 16 BY 16
938    1063 2      DO
939    1064 2      BEGIN ! DO THE CONVERSION ON BOTH WORDS SEPARATELY.
940    1065 2
941    1066 2      W = .OPWORD<.K,16>;
942    1067 2
943    1068 3      DECR I FROM 2 TO 0 DO
944    1069 4      BEGIN
945    1070 4      J = .W/.DIVTAB[I]; W = .W - .J*.DIVTAB[I];
946    1071 4      IF .J NEQ 0
947    1072 4      THEN
948    1073 5      BEGIN
949    1074 5      IF .J NEQ %0'33'
950    1075 5      THEN
951    1076 6      BEGIN
952    1077 6      IF .J LSS %0'33' THEN J = .J + %0'56';
953    1078 6      J = .J + %0'11';
954    1079 5      END;
955    1080 5      J = .J + %0'11';
956    1081 5      END
957    1082 4      ELSE
958    1083 4      J = %0'40';
959    1084 4
960    1085 4      ! AT THIS POINT, A SINGLE char IS IN BYTE 0 OF J.
961    1086 4      ! Stuff the character back and tally up the
962    1087 4      ! number of non-blank ones.
963    1088 4
964    1089 5      IF( ((.PTR)<0,8> = .J) NEQ %0'40' )
965    1090 4      then
966    1091 4      non_blanks = .non_blanks +1;
967    1092 4
968    1093 4      PTR = .PTR + 1;
969    1094 3      END;
970    1095 3
971    1096 2      END;
972    1097 2      ! END OF K LOOP.
973    1098 2
974    1099 2      ! Return the number of non-blank characters
975    1100 2      ! we stuffed back.
976    1101 2
977    1102 2      RETURN(.non_blanks);
978    1103 1      END; ! OF conv_r_50 ROUTINE.

```

.PSECT DBGSPPLIT,NOWRT, SHR, PIC,0

00000640 00000028 00000001 0008D 00090 P.AAL: .BLKB 3  
.LONG 1, 40, 1600

DIVTAB= P.AAL

						PSECT	DBGSCODE, NOWRT, SHR, PIC.O	
55	04	AC	57	00000000	08	00FC	00000	.ENTRY
	54		EF	9E	00002	MOVAB	DBGSCONV R_50, Save R2,R3,R4,R5,R6,R7	: 0999
			AC	D0	00009	MOVL	DIVTAB, R7	: 1059
			56	D4	0000D	CLRL	DST_PTR, PTR	: 1060
			53	D4	0000F	CLRL	NON_BLANKS	: 1062
			53	EF	00011	K		: 1066
	51		02	D0	00017	EXTZV	K, #16, OPWORD, W	: 1068
	50		6741	C7	0001A	MOVL	#2, I	: 1070
	52		6741	C5	0001F	DIVL3	DIVTAB[I], W, J	: 1071
	55		52	C2	00024	MULL3	DIVTABLE[I], J, R2	: 1074
			50	D5	00027	SUBL2	R2, W	: 1077
			12	13	00029	TSTL	J	: 1080
	1B		50	D1	0002B	BEQL	5\$	: 1083
			08	13	0002E	CMPL	J, #27	: 1089
			03	18	00030	BEQL	4\$	: 1091
	50		2E	C0	00032	BGEQ	3\$	: 1093
	50		09	C0	00035	ADDL2	#46, J	: 1098
	50		09	C0	00038	ADDL2	#9, J	: 1068
			03	11	0003B	ADDL2	#9, J	: 1062
	50		20	D0	0003D	BRB	6\$	: 1071
	64		50	90	00040	5\$:	MOVL	: 1083
	20		50	D1	00043	MOVB	J, (PTR)	: 1089
			02	13	00046	CMPL	J, #32	: 1091
			56	D6	00048	BEQL	7\$	: 1093
			54	D6	0004A	INCL	NON_BLANKS	: 1068
FBC	53	CB	51	F4	0004C	INCL	PTR	: 1062
	10		10	F1	0004F	SOBGEQ	I, 2\$	: 1102
	50		56	D0	00055	ACBL	#16, #16, K, 1\$	: 1103
			04	00058		MOVL	NON_BLANKS, R0	
						RET		

; Routine Size: 89 bytes, Routine Base: DBGSCODE + 04C6

```
980      1 global routine dbg$octal_valtostg_cvt(value, count) =  
981      1  
982      1    ++  
983      1    Functional Description:  
984      1    Convert a value to an ascii string. The string, when  
985      1    printed, displays the octal representation of the value.  
986      1  
987      1    Inputs  
988      1    value - the actual value we are to convert.  
989      1    count - the number of characters in the result string.  
990      1  
991      1    Routine Value  
992      1    A pointer to a counted string.  
993      1    Side effects  
994      1    Allocates space for the result. This should be  
995      1    freed by the caller when he is finished with it.  
996      1  
997      1    --  
998      2  
999      2  
1000     2  
1001     2  
1002     2  
1003     2  
1004     2  
1005     2  
1006     2  
1007     2  
1008     2  
1009     2  
1010     2  
1011     2  
1012     2  
1013     2  
1014     2  
1015     2  
1016     2  
1017     2  
1018     2  
1019     2  
1020     2  
1021     2  
1022     2  
1023     2  
1024     2  
1104     1  
1105     1  
1106     1  
1107     1  
1108     1  
1109     1  
1110     1  
1111     1  
1112     1  
1113     1  
1114     1  
1115     1  
1116     1  
1117     1  
1118     1  
1119     1  
1120     1  
1121     1  
1122     2  
1123     2  
1124     2  
1125     2  
1126     2  
1127     2  
1128     2  
1129     2  
1130     2  
1131     2  
1132     2  
1133     2  
1134     2  
1135     2  
1136     2  
1137     2  
1138     2  
1139     2  
1140     2  
1141     2  
1142     2  
1143     2  
1144     2  
1145     2  
1146     2  
1147     2  
1148     1  
  
    own  
    result_ptr : ref vector[,byte],  
    tran_table : vector[8,byte]  
                initial( byte( %ascii '01234567' ));  
  
    ! allocate space for the string.  
    result_ptr = dbg$get_memory(((1+.count)/%upval)+1);  
  
    if .result_ptr eql 0 then signal(dbgs$nofree)  
    else  
        ! fill in result string from right to left.  
        decr i from .count to 1 do  
            begin  
            result_ptr[.i] = tran_table[.value mod 8];  
            value = .value / 8  
            end;  
  
        ! fill in the count.  
        result_ptr[0] = .count;  
  
.result_ptr  
  
end: ! of dbg$octal_valtostg_cvt
```

.PSECT DBG\$OWN,NOEXE, PIC.2

00000 RESULT\_PTR:

.BLKB 4

37 36 35 34 33 32 31 30 00004 TRAN\_TABLE:  
.ASCII \01234567\

				.PSECT	DBG\$CODE, NOWRT, SHR, PIC, 0	
				.ENTRY	DBG\$OCTAL VALTOSTG_CVT, Save R2,R3	: 1104
				MOVAB	RESULT_PTR, R3	
				ADDL3	#1. COUNT, R0	: 1131
				DIVL2	#4, R0	
				PUSHAB	1(R0)	
				CALLS	#1. DBG\$GET_MEMORY	
				MOVL	R0, RESULT_PTR	
				MOVL	RESULT_PTR, R2	: 1133
				BNEQ	1\$	
				PUSHL	#164658	
				CALLS	#1. LIB\$SIGNAL	
				BRB	4\$	
				ADDL3	#1. COUNT, I	: 1137
				BRB	3\$	
				EMUL	#1. VALUE, #0, -(SP)	: 1139
				EDIV	#8, (SP)+, R0, R0	
				MOVB	TRAN_TABLE[R0], (I)[R2]	
				DIVL2	#8, VALUE	: 1140
				S0BGTR	I, 2\$	
				MOVB	COUNT, @RESULT_PTR	: 1144
				MOVL	RESULT_PTR, R0	: 1148
				RET		

: Routine Size: 90 bytes,    Routine Base: DBG\$CODE + 051F

: 1025            1149 1

```
1027      1150 1 global routine dbg$decimal_valtostg_cvt(in_value_ptr,len) =  
1028      1151 1 ++  
1029      1152 1 Functional description:  
1030      1153 1 Converts a value to an ascii string to be printed.  
1031      1154 1 Inputs  
1032      1155 1     in_value_ptr - points to the place in memory where the  
1033      1156 1             value is stored.  
1034      1157 1     len - length in bytes of the value  
1035      1158 1 Routine value  
1036      1159 1     A pointer to a counted string with the result.  
1037      1160 1     Storage for the result string is allocated dynamically  
1038      1161 1             by calling dbg$get_memory.  
1039      1162 1 Side effects  
1040      1163 1     Allocates space for the result. This should be  
1041      1164 1             released by the caller.  
1042      1165 1 --  
1043      1166 2 begin  
1044      1167 2 map  
1045      1168 2     in_value_ptr : ref bitvector [] ;  
1046      1169 2  
1047      1170 2 local  
1048      1171 2     value_copy : bitvector[128],  
1049      1172 2     value_ptr : ref bitvector[],  
1050      1173 2     sign_flag,  
1051      1174 2     string : ref vector[,byte],  
1052      1175 2     power_of_two,  
1053      1176 2     new_string : ref vector[,byte],  
1054      1177 2     new_power_of_two;  
1055      1178 2  
1056      1179 2  
1057      1180 2  
1058      1181 2  
1059      1182 2  
1060      1183 2  
1061      1184 2  
1062      1185 2  
1063      1186 2  
1064      1187 2  
1065      1188 2  
1066      1189 2  
1067      1190 2  
1068      1191 2  
1069      1192 2  
1070      1193 2  
1071      1194 2  
1072      1195 2  
1073      1196 2  
1074      1197 2  
1075      1198 2  
1076      1199 2  
1077      1200 2  
1078      1201 2  
1079      1202 2  
1080      1203 2  
1081      1204 2  
1082      1205 2  
1083      1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2  
1168 2  
1169 2  
1170 2  
1171 2  
1172 2  
1173 2  
1174 2  
1175 2  
1176 2  
1177 2  
1178 2  
1179 2  
1180 2  
1181 2  
1182 2  
1183 2  
1184 2  
1185 2  
1186 2  
1187 2  
1188 2  
1189 2  
1190 2  
1191 2  
1192 2  
1193 2  
1194 2  
1195 2  
1196 2  
1197 2  
1198 2  
1199 2  
1200 2  
1201 2  
1202 2  
1203 2  
1204 2  
1205 2  
1206 2  
1150 1  
1151 1  
1152 1  
1153 1  
1154 1  
1155 1  
1156 1  
1157 1  
1158 1  
1159 1  
1160 1  
1161 1  
1162 1  
1163 1  
1164 1  
1165 1  
1166 2  
1167 2
```

```

1084    1207   6
1085    1208   6
1086    1209   6
1087    1210   4
1088    1211   4
1089    1212   3
1090    1213   3
1091    1214   4
1092    1215   4
1093    1216   5
1094    1217   6
1095    1218   6
1096    1219   5
1097    1220   4
1098    1221   4
1099    1222   3
1100    1223   3
1101    1224   4
1102    1225   4
1103    1226   4
1104    1227   4
1105    1228   3
1106    1229   6
1107    1230   4
1108    1231   4
1109    1232   4
1110    1233   3
1111    1234   3
1112    1235   2

        .b[i-(n-m)]
else .b[i] +
.a[i-(n-m)])
- 2*%C'0');
carry = .temp geq 10;
end;
decr i from .n-m to 1 do
begin
result[i+1] = .ctable[temp =
(.carry +
(if .a[0] gtr .b[0]
then .a[i]
else .b[i]) -
%C'0'));
carry = .temp geq 10;
end;
if .carry eql 1 then
begin
result[0] = .n+1;
result[1] = %C'1';
end
else
begin
result[0] = .n;
ch$move(.n,
result[2],result[1]);
end;
.result
end; ! addc

```

## .PSECT DBG\$OWN,NOEXE, PIC,2

34	33	32	31	30	39	38	37	36	35	34	33	32	31	30	0000C CTABLE: .ASCII \01234567890123456789\
39	38	37	36	35											

## .PSECT DBG\$CODE,NOWRT, SHR, PIC,0

			07FC 00000 ADDC:	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	1182	
		5A 00000000'	EF 9E 00002	MOVAB	CTABLE, R10		
08	BC	04	58 D4 00009	CLRL	R8	1195	
			BC 91 0000B	CMPB	AA, BB		
		08	1B 00010	BLEQU	1S		
		52	04	58 D6 00012	INCL	R8	
		04	BC 9A 00014	MOVZBL	AA, N		
		52	08	BC 9A 0001A	BRB	2S	
		06	04	11 00018	MOVZBL	BB, N	
		52	08	BC 9A 0001E	BLBC	R8, 3S	1197
		04	58 F9 00021	MOVZBL	BB, M		
		54	08	BC 9A 00021	BRB	4S	
		54	04	04 11 00025	MOVZBL	AA, M	
		50	02	A2 9E 0002B	MOVAB	2(R2), R6	
		50	04	F6 0002F	DIVL2	#4, R6	
		01	A0 9F 00032	PUSHAB	1(R0)	1199	

	00000000G	00	01	FB	00035	CALLS	#1, DBG\$GET_MEMORY	
	56	56	50	D0	0003C	MOVL	R0, RESULT	1200
50	52	52	53	D4	0003F	CLRL	CARRY	1201
55	57	57	54	C3	00041	SUBL3	M, N, R0	1207
	52	52	A0	9E	00045	MOVAB	1(R0), R7	
	50	50	54	C3	00049	SUBL3	M, N, R5	
	52	52	D0	0004D	MOVL	N, I		
51	50	59	11	00050	BRB	9\$		
	0C	55	C3	00052	5\$: SUBL3	R5, I, R1		
	59	58	E9	00056	BLBC	R8, 6\$, R1		
51	51	04 BC40	9A	00059	MOVZBL	0A[I], R9	1209	
	08 BC41	9A	0005E	MOVZBL	0B[R1], R1			
	0A	0A	11	00063	BRB	7\$		
59	08 BC40	9A	00065	6\$: MOVZBL	0B[I], R9			
51	04 BC41	9A	0006A	MOVZBL	0A[R1], R1	1204		
51	59	C0	0006F	7\$: ADDL2	R9, R1	1203		
54	A0 A143	9E	00072	MOVAB	-96(R1)[CARRY], TEMP	1211		
	6A44	90	00077	MOVB	CTABLE[TEMP], {I}[RESULT]			
01 A046	51	D4	0007D	CLRL	R1			
	51	D1	0007F	CMPL	TEMP, #10			
	0A	02	19	00082	BLSS	8\$		
	54	D6	00084	INCL	R1			
53	51	D0	00086	MOVL	R1, CARRY	1201		
	50	50	D7	00089	DECL	I		
57	50	D1	0008B	CMPL	I, R7	1213		
	C2	18	0008E	BGEQ	5\$			
50	01 A5	9E	00090	MOVAB	1(R5), I	1217		
	26	11	00094	BRB	14\$			
07	58	E9	00096	10\$: BLBC	R8, 11\$	1218		
51	04 BC40	9A	00099	MOVZBL	0A[I], R1			
	05	11	0009E	BRB	12\$			
51	08 BC40	9A	000A0	11\$: MOVZBL	0B[I], R1	1219		
54	D0 A143	9E	000A5	12\$: MOVAB	-48(R1)[CARRY], TEMP	1216		
	6A44	90	000AA	MOVB	CTABLE[TEMP], {I}[RESULT]	1215		
01 A046	51	D4	000B0	CLRL	R1	1221		
	51	D1	000B2	CMPL	TEMP, #10			
	0A	02	19	000B5	BLSS	13\$		
	54	D6	000B7	INCL	R1			
53	51	D0	000B9	13\$: MOVL	R1, CARRY	1213		
07	50	F5	000BC	14\$: SOBGTR	I, 10\$	1223		
01	53	D1	000BF	CMPL	CARRY, #1			
66	01 52	01	81	000C4	BNEQ	15\$		
	A6	31	90	000C8	ADDB3	#1, N, (RESULT)	1225	
	02 66	09	11	000CC	MOVB	#49, 1(RESULT)	1226	
01 A6	52	90	000CE	BRB	16\$	1223		
	52	28	000D1	15\$: MOVB	N, (RESULT)	1230		
	50	D0	000D7	MOVCF3	N, 2(RESULT), 1(RESULT)	1232		
	56	D0	000DA	MOVL	RESULT, R0	1235		
	04	000DA	RET					

: Routine Size: 219 bytes.    Routine Base: DBGS\$CODE + 0579

:	1113	1236	2
:	1114	1237	2
:	1115	1238	2
:	1116	1239	2

| Copy the value to be examined into a local variable  
value\_ptr = value\_copy;

```
1117      1240 2
1118      1241 2
1119      1242 2
1120      1243 2
1121      1244 2
1122      1245 2
1123      1246 2
1124      1247 2
1125      1248 2
1126      1249 2
1127      1250 2
1128      1251 2
1129      1252 2
1130      1253 2
1131      1254 2
1132      1255 2
1133      1256 2
1134      1257 2
1135      1258 2
1136      1259 2
1137      1260 2
1138      1261 2
1139      1262 3
1140      1263 4
1141      1264 4
1142      1265 4
1143      1266 4
1144      1267 4
1145      1268 4
1146      1269 3
1147      1270 3
1148      1271 3
1149      1272 3
1150      1273 3
1151      1274 3
1152      1275 3
1153      1276 3
1154      1277 3
1155      1278 3
1156      1279 3
1157      1280 3
1158      1281 3
1159      1282 3
1160      1283 3
1161      1284 3
1162      1285 3
1163      1286 3
1164      1287 3
1165      1288 3
1166      1289 3
1167      1290 3
1168      1291 3
1169      1292 3
1170      1293 2
1171      1294 2
1172      1295 1

ch$move (16, .in_value_ptr, .value_ptr);

! now build up print string representing the
! octaword integer.
string = dbg$get_memory(1);
power_of_two = dbg$get_memory(1);
(.string) = %ASCII('0');
(.power_of_two) = %ASCII('1');
sign_flag = 0;
IF .value_ptr[8*.len-1] EQL 1
THEN ! sign bit set.
BEGIN
sign_flag = 1;
! negate number.
incr i from 0 to .len/4-1 do
(.value_ptr+4*i) =
NOT (.value_ptr+4*i);
END;
incr i from 0 to (8*.len)-2 do
begin
! look at the ith bit of the integer.
IF .value_ptr[i] EQL 1
THEN
BEGIN
! if the bit is set, must add in the
! appropriate power of two.
new_string = addc(.string,.power_of_two);
dbg$rel_memory(.string);
string = .new_string;
END;
new_power_of_two = addc(
.power_of_two,.power_of_two);
dbg$rel_memory(.power_of_two);
power_of_two = .new_power_of_two;
end; ! loop
IF .sign_flag EQL 1
THEN
BEGIN
! un-complement number
! (we complemented earlier)
incr i from 0 to .len/4-1 do
(.value_ptr+4*i) =
NOT (.value_ptr+4*i);
! add 1 (we complemented earlier)
new_string = addc(.string,
"UPLIT BYTE(%ASCII '1')");
! append minus sign.
string = dbg$get_memory(1+(2+.new_string[0])/4);
string[0] = 1+.new_string[0];
string[1] = %( '-';
ch$move(.new_string[0],new_string[1],string[2]);
dbg$rel_memory(.new_string);
END;
dbg$rel_memory(.power_of_two);
.string
end; ! of dbg$decimal_valtostg_cvt
```

					.PSECT	DBG\$PLIT,NOWRT, SHR, PIC,O	
			31 01 0009C P.AAM:		.ASCII	<1>\1\	
					.PSECT	DBG\$CODE,NOWRT, SHR, PIC,O	
					.ENTRY	DBG\$DECIMAL_VALTOSTG_CVT, Save R2,R3,R4,R5,-; 1150	
					MOVAB	R6,R7,R8,R9-R10,R11	
					SUBL2	DBG\$REL_MEMORY, R11	
					MOVAB	#16, SP	
					MOVC3	VALUE COPY, VALUE_PTR	
					PUSHL	#16, \$IN_VALUE_PTR, (VALUE_PTR)	
					CALLS	#1, DBG\$GET_MEMORY	
					MOVL	R0, STRING	
					PUSHL	#1	
					CALLS	#1, DBG\$GET_MEMORY	
					MOVL	R0, POWER_OF_TWO	
					MOVZWL	#12289, (STRING)	
					MOVZWL	#12545, (POWER_OF_TWO)	
					CLRL	SIGN_FLAG	
					MOVL	LEN, R3	
					ASHL	#3, R3, R0	
					DECL	R0	
					EXTZV	R0, #1, (VALUE_PTR), R1	
					CMPL	R1, #1	
					BNEQ	38	
					MOVL	#1, SIGN_FLAG	
					DIVL3	#4, R3, R4	
					MNEGL	#1, I	
					BRB	28	
					MCOML	(VALUE_PTR)[I], (VALUE_PTR)[I]	
					AOBLSS	R4, I 18	
					ASHL	#3, R3, R4	
					SUBL2	#2, R4	
					MNEGL	#1, I	
					BRB	68	
					EXTZV	I, #1, (VALUE_PTR), R0	
					CMPL	R0, #1	
					BNEQ	58	
					PUSHR	#^H<R6,R9>	
					CALLS	#2, ADDC	
					MOVL	R0, NEW_STRING	
					PUSHL	STRING	
					CALLS	#1, DBG\$REL_MEMORY	
					MOVL	NEW_STRING, STRING	
					PUSHL	POWER_OF_TWO	
					PUSHL	POWER_OF_TWO	
					CALLS	#2, ADDC	
					MOVL	R0, NEW_POWER_OF_TWO	
					PUSHL	POWER_OF_TWO	
					CALLS	#1, DBG\$REL_MEMORY	
					MOVL	NEW_POWER_OF_TWO, POWER_OF_TWO	

CA	52	54	F3	0009F	68:	AOBLEQ	R4, I, 48	1258	
	01	55	D1	000A3		CMPL	SIGN_FLAG, #1	1275	
	53	4D	12	000A6		BNEQ	98		
	50	04	C0	000A8		DIVL2	#4, R3	1280	
	05	01	CE	000AB		MNEGL	#1, I		
	6740	05	11	000AE		BRB	88		
F7	6740	6740	D2	000B0	78:	MCOML	(VALUE_PTR)[I], (VALUE_PTR)[I]	1282	
	50	53	F2	000B5	88:	AOBLSS	R3, I, -78	1281	
	000000000.	EF	9F	000B9		PUSHAB	P.AAM	1285	
	FESF	56	DD	000BF		PUSHL	STRING	1284	
	CF	02	FB	000C1		CALLS	#2, ADDC		
	58	50	D0	000C6		MOVL	R0, NEW STRING		
	50	68	9A	000C9		MOVZBL	(NEW STRING), R0	1287	
	50	02	CO	000CC		ADDL2	#2, R0		
	50	04	C6	000CF		DIVL2	#4, R0		
	00000000G	01	A0	9F	000D2	PUSHAB	1(R0)		
	00	01	FB	000D5		CALLS	#1, DBG\$GET_MEMORY		
	56	50	D0	000DC		MOVL	R0, STRING		
66	68	01	81	000DF		ADDB3	#1, (NEW STRING), (STRING)	1288	
	01	A6	2D	90	000E3	MOVB	#45, 1(STRING)	1289	
	50	68	9A	000E7		MOVZBL	(NEW STRING), R0	1290	
02	A6	01	A8	50	28	000EA	MOVC3	R0, T(NEW STRING), 2(STRING)	
	68	58	DD	000FO		PUSHL	NEW_STRING	1291	
	68	01	FB	000F2		CALLS	#1, DBGSREL MEMORY		
	68	59	DD	000F5	98:	PUSHL	POWER OF TWO	1293	
	50	01	FB	000F7		CALLS	#1, DBGSREL_MEMORY		
	56	04	DD	000FA		MOVL	STRING, R0	1295	
						RET			

; Routine Size: 254 bytes,    Routine Base: DBGSCODE + 0654

```
1174 1296 1 | THE REGISTER TABLE HOLDS ONE ENTRY PER REGISTER. EACH ENTRY IS MADE
1175 1297 1 | UP OF ONE LONGWORD. THE FIRST BYTE HOLDS THE CHARACTER COUNT OF
1176 1298 1 | THE REGISTER NAME. THE SECOND THROUGH FOURTH BYTES HOLD THE REGISTER
1177 1299 1 | NAME STRING. A SAMPLE ENTRY FOLLOWS:
1178 1300 1 |
1179 1301 1 |
1180 1302 1 |
1181 1303 1 |
1182 1304 1 |
1183 1305 1 |
1184 1306 1 |
1185 M 1307 1 | MACRO
1186 1308 1 | register_entry (string) =
1187 1309 1 | %CHARCOUNT (STRING), %ASCII STRING, REP 3 - %CHARCOUNT (STRING) OF BYTE (0)%;
1188 1310 1 |
1189 1311 1 | BIND
1190 1312 1 | register_table = UPLIT BYTE (
1191 1313 1 |     register_entry ('R0'),
1192 1314 1 |     register_entry ('R1'),
1193 1315 1 |     register_entry ('R2'),
1194 1316 1 |     register_entry ('R3'),
1195 1317 1 |     register_entry ('R4'),
1196 1318 1 |     register_entry ('R5'),
1197 1319 1 |     register_entry ('R6'),
1198 1320 1 |     register_entry ('R7'),
1199 1321 1 |     register_entry ('R8'),
1200 1322 1 |     register_entry ('R9'),
1201 1323 1 |     register_entry ('R10'),
1202 1324 1 |     register_entry ('R11'),
1203 1325 1 |     register_entry ('AP'),
1204 1326 1 |     register_entry ('FP'),
1205 1327 1 |     register_entry ('SP'),
1206 1328 1 |     register_entry ('PC'),
1207 1329 1 |     register_entry ('PSL'));
1208 1330 1 |
1209 1331 1 | BLOCK [, LONG];
1210 1332 1 |
1211 1333 1 | ++
1212 1334 1 | THESE FIELD DEFINITIONS CONTROL ACCESS TO THE REGISTER TABLE
1213 1335 1 | --
1214 1336 1 | MACRO
1215 1337 1 |     REG_NAME      =8, 24, 0%,
1216 1338 1 |     ctd_reg_name =0, 24, 0%,
1217 1339 1 |     REG_CH_CNT   =0, 8, 0%:
1218 1340 1 |
1219 1341 1 | ++
1220 1342 1 | COMMON ASCII COUNTED STRINGS USED IN FAO CALLS.
1221 1343 1 | --
1222 1344 1 |
1223 1345 1 | BIND
1224 1346 1 |     cs_ascii      = UPLIT ( %ASCIC '!AD'),
1225 1347 1 |     colon_tab_stg = UPLIT ( %ASCIC ':');
1226 1348 1 |
1227 1349 1 | ++
1228 1350 1 | The following macros translate addresses to register offsets and
1229 1351 1 | vice versa.
1230 1352 1 | --
```

: 1231		1353	1
.. 1232	M	1354	1
.. 1233	M	1355	1
.. 1234		1356	1
.. 1235		1357	1
.. 1236	M	1358	1
: 1237		1359	1

## MACRO

```
this_is_reg (location) =  
    (((location) GEQA dbg$runframe [dbg$1_user_r0]) AND  
     ((location) LEQA dbg$runframe [dbg$1_user_ps1]))%,  
  
reg_offset (location) =  
    (location - dbg$runframe [dbg$1_user_regs]) / %UPVAL%;
```

```
1239      1360 1 GLOBAL ROUTINE dbg$out_regname( address ) =
1240          1361 1 ++
1241          1362 1 Functional Description
1242          1363 1
1243          1364 1     Given an address, see if it falls within the current
1244          1365 1     runframe in such a way as one could say that this address
1245          1366 1     corresponds to one of the general registers. If this is
1246          1367 1     not the case, we return FALSE. Otherwise we output the
1247          1368 1     name of the indicated register and return TRUE.
1248          1369 1
1249          1370 1 Formal Parameters:
1250          1371 1
1251          1372 1     address -the address which we are trying to symbolize
1252          1373 1 Implicit Inputs:
1253          1374 1
1254          1375 1     The format and use of the register table
1255          1376 1     which is local to this module.
1256          1377 1
1257          1378 1 Return Value
1258          1379 1     TRUE or FALSE. See above.
1259          1380 1
1260          1381 1 Side Effects:
1261          1382 1
1262          1383 1     We may output a register name to the output buffer.
1263          1384 1
1264          1385 1 --
1265          1386 1 BEGIN
1266          1387 2
1267          1388 2     BIND      register_vector = dbg$runframe [ DBGSL_USER_REGS ] : VECTOR;
1268          1389 2
1269          1390 2     LOCAL    reg_index;
1270          1391 2
1271          1392 2     IF( NOT this_is_reg(.address) )
1272          1393 2     then      return(false);
1273          1394 2
1274          1395 2     reg_index = reg_offset(.address);
1275          1396 2
1276          1397 2     ! Check that the address EXACTLY matches one which
1277          1398 2     ! we currently bind to a register name.
1278          1399 2
1279          1400 2     IF( register_vector[.reg_index] NEQA .address )
1280          1401 2     then      return(FALSE);
1281          1402 2
1282          1403 2
1283          1404 2
1284          1405 2     ! An exact match has been found.
1285          1406 2     ! Output the register name and return
1286          1407 2     ! a success status.
1287          1408 2
1288          1409 2     dbgSprint( UPLIT( %ASCIC '!AC' ),
1289          1410 2             register_table[.reg_index, ctd_reg_name ] );
1290          1411 2
1291          1412 2
1292          1413 2 END:
```

.PSECT DBGSPLIT,NOWRT, SHR, PIC,0

30	02	0009E	P.AAN:	.BYTE	2
	52	0009F		.ASCII	\R0\
	00	000A1		.BYTE	0
31	02	000A2		.BYTE	2
	52	000A3		.ASCII	\R1\
	00	000A5		.BYTE	0
32	02	000A6		.BYTE	2
	52	000A7		.ASCII	\R2\
	00	000A9		.BYTE	0
33	02	000AA		.BYTE	2
	52	000AB		.ASCII	\R3\
	00	000AD		.BYTE	0
34	02	000AE		.BYTE	2
	52	000AF		.ASCII	\R4\
	00	000B1		.BYTE	0
35	02	000B2		.BYTE	2
	52	000B3		.ASCII	\R5\
	00	000B5		.BYTE	0
36	02	000B6		.BYTE	2
	52	000B7		.ASCII	\R6\
	00	000B9		.BYTE	0
37	02	000BA		.BYTE	2
	52	000BB		.ASCII	\R7\
	00	000BD		.BYTE	0
38	02	000BE		.BYTE	2
	52	000BF		.ASCII	\R8\
	00	000C1		.BYTE	0
39	02	000C2		.BYTE	2
	52	000C3		.ASCII	\R9\
	00	000C5		.BYTE	0
30	03	000C6		.BYTE	3
	31	52	000C7	.ASCII	\R10\
31	03	000CA		.BYTE	3
	31	52	000CB	.ASCII	\R11\
	02	000CE		.BYTE	2
50	41	000CF		.ASCII	\AP\
	00	000D1		.BYTE	0
50	02	000D2		.BYTE	2
	46	000D3		.ASCII	\FP\
	00	000D5		.BYTE	0
50	02	000D6		.BYTE	2
	53	000D7		.ASCII	\SP\
	00	000D9		.BYTE	0
43	02	000DA		.BYTE	2
	50	000DB		.ASCII	\PC\
	00	000DD		.BYTE	0
40	03	000DE		.BYTE	3
	53	50	000DF	.ASCII	\PSL\
	00	000E2		.BLKB	2
44	41	21	03	P.AAO:	.ASCII <3>\!AD\
20	20	3A	03	P.AAP:	.ASCII <3>\!: \
43	41	21	03	P.AAQ:	.ASCII <3>\!AC\

REGISTER TABLE = P.AAN  
CS\_ASCII = P.AAO

## COLON\_TAB\_STG= P.AAP

			.PSECT	DBG\$CODE, NOWRT, SHR, PIC, 0	
			.ENTRY	DBG\$OUT REGNAME, Save R2	: 1360
		52 00000000G	00 9E 00002	DBG\$RUNFRAME+4, R2	
		50 04	62 9E 00009	DBG\$RUNFRAME+4, R0	: 1392
		50 04	AC D1 0000C	CMPL ADDRESS, R0	
		50 40	37 1F 00010	BLSU 1\$	
		50 04	A2 9E 00012	MOVAB DBG\$RUNFRAME+68, R0	
		50 04	AC D1 00016	CMPL ADDRESS, R0	
		50 04	2D 1A 0001A	BGTRU 1\$	
		50 04 AC	62 9E 0001C	MOVAB DBG\$RUNFRAME+4, R0	: 1396
		50 04 AC	50 C3 0001F	SUBL3 R0, ADDRESS, R0	
		50 04 AC	04 C6 00024	DIVL2 #4, REG INDEX	
		51 04 AC	6240 DE 00027	MOVAL REGISTER_VECTOR[REG_INDEX], R1	: 1401
		51 04 AC	51 D1 0002B	CMPL R1, ADDRESS	
		51 04 AC	18 12 0002F	BNEQ 1\$	
		00000000' 00000000'	EF40 DF 00031	PUSHAL REGISTER_TABLE[REG_INDEX]	: 1410
		00000000' 00000000'	EF 9F 00038	PUSHAB P.AAQ	: 1409
		00000000G 00 50	02 FB 0003E	CALLS #2, DBG\$PRINT	: 1410
		00000000G 00 50	01 D0 00045	MOVL #1, R0	: 1412
		00000000G 00 50	04 00048	RET	
		00000000G 00 50	50 D4 00049	CLRL R0	: 1413
		00000000G 00 50	1\$: 04 0004B	RET	

; Routine Size: 76 bytes,   Routine Base: DBG\$CODE + 0752

```

1294      1414 1 GLOBAL ROUTINE dbg$reg_match (string_desc) =
1295      1415 1 ++
1296      1416 1 Functional description:
1297      1417 1   Compares a string described by the string descriptor passed
1298      1418 1   as the routine formal to each of the names of the machine
1299      1419 1   registers. If the string matches a register name, return the
1300      1420 1   number of the register (0-16, where 16 is the PSL). Otherwise,
1301      1421 1   return a -1.
1302      1422 1
1303      1423 1 Inputs:
1304      1424 1   string_desc - string descriptor to symbol string
1305      1425 1
1306      1426 1 Implicit inputs:
1307      1427 1   the VAX machine register table
1308      1428 1
1309      1429 1 Implicit outputs:
1310      1430 1   none
1311      1431 1
1312      1432 1 Routine value:
1313      1433 1   The register number 0 - 16 if a match is found.
1314      1434 1   -1 if no match is found.
1315      1435 1
1316      1436 1 Side effects:
1317      1437 1   none
1318      1438 1 --
1319      1439 1
1320      1440 2 BEGIN
1321      1441 2
1322      1442 2 MAP
1323      1443 2   string_desc : REF BLOCK [, BYTE];
1324      1444 2
1325      1445 2
1326      1446 2
1327      1447 2 INCR index from 0 to register_count-1 DO
1328      1448 3 BEGIN
1329      1449 3   IF ch$eq1 (.string_desc[dsc$w_length], ch$ptr(.string_desc [dsc$sa_pointer]),
1330      1450 3       .register_table[index, reg_ch_cnt],
1331      1451 3       ch$ptr(register_table[index, reg_name]))
1332      1452 3 THEN
1333      1453 3   RETURN .index;
1334      1454 2 END;
1335      1455 2
1336      1456 2 RETURN (-1);
1337      1457 2
1338      1458 1 END;

```

50 00 04 B5 04 BC 2D 00019	003C 00000	55 04 AC D0 00002	54 D4 00006	50 00000000'EF44 DF 00008	1\$: 9E 9A 0000F	50 00000000'EF44 DF 00012	0: 04 BC 2D 00019	ENTRY DBGSREG_MATCH, Save R2,R3,R4,R5	MOVL STRING_DESC, R5	CLRL INDEX	PUSHAL REGISTER_TABLE[INDEX]	A(SP)+, R0	PUSHAL REGISTER_TABLE+1[INDEX]	ASTRING_DESC, A4(R5), #0, R0, A(SP)+
----------------------------	------------	-------------------	-------------	---------------------------	------------------	---------------------------	-------------------	---------------------------------------	----------------------	------------	------------------------------	------------	--------------------------------	--------------------------------------

: 1414  
 : 1449  
 : 1450  
 : 1451

	9E	00020		
	04	12 00021	BNEQ	2\$
	54	D0 00023	MOVL	INDEX, R0
		C4 00026	RET	
DD	54	10 F3 00027 2\$:	AOBLEQ	#16, INDEX, 1\$
	50	01 CE 00028	MNEG.L	#1, R0
		04 0002E	RET	

1453

1447

1456

1458

: Routine Size: 47 bytes.    Routine Base: DBGSCODE + 079E

```
1340      1459 1 GLOBAL ROUTINE dbg$output_psl (value) : NOVALUE =
1341      1460 1 ++
1342      1461 1 FUNCTIONAL DESCRIPTION:
1343      1462 1     Formats and outputs two lines of specially
1344      1463 1     formatted data contained in the PSL. The fields shown are
1345      1464 1
1346      1465 1     CMP      - compatibility mode
1347      1466 1     TP       - trace trap pending
1348      1467 1     FPD      - first part done
1349      1468 1     IS       - interrupt stack
1350      1469 1     CURMOD   - current access mode
1351      1470 1     PRVMOD   - previous access mode
1352      1471 1     IPL      - interrupt priority level
1353      1472 1     DV       - decimal overflow trap enable
1354      1473 1     FU       - floating underflow trap enable
1355      1474 1     IV       - integer overflow trap enable
1356      1475 1     T        - trace trap
1357      1476 1     N        - condition code
1358      1477 1     Z        - condition code
1359      1478 1     V        - condition code
1360      1479 1     C        - condition code
1361      1480 1
1362      1481 1 CALLING SEQUENCE:
1363      1482 1     dbg$output_psl ()
1364      1483 1
1365      1484 1 INPUTS:
1366      1485 1     value   -The current contents of the PSL
1367      1486 1
1368      1487 1 IMPLICIT INPUTS:
1369      1488 1     NONE
1370      1489 1
1371      1490 1 OUTPUTS:
1372      1491 1     NONE
1373      1492 1
1374      1493 1 IMPLICIT OUTPUTS:
1375      1494 1     NONE
1376      1495 1
1377      1496 1 ROUTINE value:
1378      1497 1     NOVALUE
1379      1498 1
1380      1499 1 SIDE EFFECTS:
1381      1500 1     Two lines are output to the output device.
1382      1501 1
1383      1502 1
1384      1503 2 BEGIN
1385      1504 2     MAP
1386      1505 2     value: BLOCK;
1387      1506 2
1388      1507 2     MACRO
1389      1508 2     position_field = 0, 8, 1%,
1390      1509 2     size_field    = 8, 8, 1%,
1391      1510 2     mode_field   = 16, 4, 1%,
1392      1511 2     blanks_field = 20, 4, 1%,
1393      1512 2     width_field  = 24, 8, 1%,
1394      1513 2
1395      1514 2     psl_field (name, position, size, mode, width, leading_blanks) =
1396      1515 2           position, size, mode OR (leading_blanks ^ 4), width%;
```

```

1397      1516 2
1398      1517 2
1399      1518 2
1400      1519 2
1401      1520 2
1402      1521 2
1403      1522 2
1404      1523 2
1405      1524 2
1406      1525 2
1407      1526 2
1408      1527 2
1409      1528 2
1410      1529 2
1411      1530 2
1412      1531 2
1413      1532 2
1414      1533 2
1415      1534 2
1416      1535 2
1417      1536 2
1418      1537 2
1419      1538 2
1420      1539 2
1421      1540 2
1422      1541 2
1423      1542 2
1424      1543 2
1425      1544 2
1426      1545 2
1427      1546 2
1428      1547 2
1429      1548 2
1430      1549 2
1431      1550 2
1432      1551 2
1433      1552 2
1434      1553 2
1435      1554 2
1436      1555 2
1437      1556 2
1438      1557 2
1439      1558 2
1440      1559 2
1441      1560 2
1442      1561 2
1443      1562 2
1444      1563 2
1445      1564 2
1446      1565 2
1447      1566 2
1448      1567 2
1449      1568 2
1450      1569 2
1451      1570 2
1452      1571 2
1453      1572 3

LITERAL
    decimal      = 0;
    max_psl_field = 15;

BIND
    psl_table = UPLIT BYTE (
        psl_field (CMP, 31, 1, 0, 1, 1),
        psl_field (TP, 30, 1, 0, 1, 3),
        psl_field (FPD, 27, 1, 0, 1, 2),
        psl_field (IS, 26, 1, 0, 1, 3),
        psl_field (CURMOD, 24, 2, 1, 4, 2),
        psl_field (PRVMOD, 22, 2, 1, 4, 3),
        psl_field (IPL, 16, 5, 0, 2, 3),
        psl_field (DV, 7, 1, 0, 1, 2),
        psl_field (FU, 6, 1, 0, 1, 2),
        psl_field (IV, 5, 1, 0, 1, 2),
        psl_field (T, 4, 1, 0, 1, 1),
        psl_field (N, 3, 1, 0, 1, 1),
        psl_field (Z, 2, 1, 0, 1, 1),
        psl_field (V, 1, 1, 0, 1, 1),
        psl_field (C, 0, 1, 0, 1, 1));
    : BLOCK,
    hex_number      = UPLIT BYTE (%ASCIC '!AD!#XB'),
    stg_desc        = UPLIT BYTE (%ASCIC '!AD!AD'),
    blanks          = UPLIT BYTE (%ASCII ' '),
    priv_modes      = UPLIT BYTE (
        %ASCII 'KERN',
        %ASCII 'EXEC',
        %ASCII 'SUPR',
        %ASCII 'USER');

    : VECTOR;
    ! Write out the standard title which describes the PSL fields.
    dbg$print( UPLIT (%ASCIC '!_CMP TP FPD IS CURMOD PRVMOD IPL DV FU IV T N Z V ()'));
    dbg$newline();
    dbg$print (UPLIT (%ASCIC '! '));
    INCR count FROM 0 TO max_psl_field - 1 DO
        IF .psl_table [.count, mode_field] EQ decimal
        THEN
            BEGIN
                dbg$print (hex_number,
                    .psl_table [.count, blanks_field], blanks,
                    .psl_table [.count, width_field],
                    .value [0, .psl_table [.count, position_field]],
                    .psl_table [.count, size_field], 0]);
            END
        ELSE
            BEGIN
                dbg$print (stg_desc,
                    .psl_table [.count, blanks_field], blanks,

```

DBGLEVEL1  
V04-000

H 4  
16-Sep-1984 01:27:02 YAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:17:02 [DEBUG.SRC]DBGLEVEL1.832;1

Page 49  
(15)

1454	1573	END:
1455	1574	
1456	1575	
1457	1576	
1458	1577	

**END:**

```
.psl_table [.count, width_field],  
priv_modes [.value [0, .psl_table [.count, position_field],  
        .psl_table [.count, size_field], 0]]);
```

PSL_TABLE=	P.AAR
HEX_NUMBER=	P.AAS
STG_DESC=	P.AAT
BLANKS=	P.AAU
PRIV_MODES=	P.AAV

.PSECT DBG\$CODE,NOWRT. SHR. PIC.O

.ENTRY DBGSOUTPUT\_PSL. Save R2,R3,R4,R5  
MOVAB DBGSPRINT\_RS  
MOVAB BLANKS, R4  
PUSHAB P\_AAW  
CALLS #1, DBGSPRINT  
CALLS #0, DBGSNEWLINE  
P\_AAX  
#1, DBGSPRINT  
COUNT  
MOVAL PSL TABLE[COUNT], R3  
CVTBL (R3), R1  
EXTZV R1, 1(R3), VALUE, R0  
BITB 2(R3), #15  
BNEQ Z8  
PUSHL R0  
CVTBL 3(R3), -(SP)  
PUSHL R4  
EXTV #20, #4, (R3), -(SP)

		F1	A6	9F	00047	PUSHAB	HEX_NUMBER	
			12	11	0004A	BRB	38	1563
		06	A440	DF	0004C	28:	PUSHAL PRIV MODES[R0]	1575
		7E	03	A3	98	00050	CVTBL 3(R3), -(SP)	1571
			54	DD	00054	PUSHL R4	1575	
7E	63	04	F9	14	FE	00056	EXTV #20, #4, (R3), -(SP)	1571
			A4	9F	0005B	PUSHAB STG_DESC	1575	
		65	05	FB	0005E	38:	CALLS #5 DBGSPRINT	1560
		CO	52	0E	F3	00061	A0BLEQ #14, COUNT, 15	1577
				04	00065	RET		

; Routine Size: 102 bytes.    Routine Base: DBGS.CODE + 07CD

```
1460 1578 1 GLOBAL ROUTINE dbg$digit_scan(a,l,nd)=  
1461 1579 1 ++  
1462 1580 1 FUNCTIONAL DESCRIPTION:  
1463 1581 1  
1464 1582 1 This routine will scan the string pointed to by 'a' with  
1465 1583 1 length 'l' to determine if it is a valid digit string.  
1466 1584 1 It will also build a new descriptor for the input  
1467 1585 1  
1468 1586 1 INPUTS:  
1469 1587 1  
1470 1588 1 a - address of string  
1471 1589 1 l - length of string  
1472 1590 1 nd - pointer to new descriptor  
1473 1591 1  
1474 1592 1 IMPLICIT INPUTS:  
1475 1593 1  
1476 1594 1  
1477 1595 1 OUTPUTS:  
1478 1596 1 none  
1479 1597 1  
1480 1598 1 IMPLICIT OUTPUTS:  
1481 1599 1  
1482 1600 1 ROUTINE VALUE:  
1483 1601 1 1 - a valid digit string  
1484 1602 1 3 - a valid digit string beginning with a sign + or -  
1485 1603 1 0 - not a valid digit string  
1486 1604 1  
1487 1605 1 SIDE EFFECTS:  
1488 1606 1  
1489 1607 2  
1490 1608 2 BEGIN  
1491 1609 2 builtin cvtsp,cvtpl;  
1492 1610 2 MAP a : ref vector[,byte],nd : ref block[,byte];  
1493 1611 2 LOCAL s,f,dp,dpp,ep,esn,p : vector [40,byte],ln,f,nwl;  
1494 1612 2 BIND max_packed_size = uplit(31);  
1495 1613 2 ! Quick fix for a problem: this routine was returning "true" on the  
1496 1614 2 string "e".  
1497 1615 2  
1498 1616 2 IF .L EQL 1 AND (.A[0] EQL 'E' OR .A[0] EQL 'e')  
1499 1617 2 THEN  
1500 1618 2 RETURN 0;  
1501 1619 2  
1502 1620 2 i = 0 ;  
1503 1621 2 s = 1 ;  
1504 1622 2 f = dp=ep=dpp=esn=0;  
1505 1623 2  
1506 1624 2 ! get possible trailing spaces  
1507 1625 2 nwl = .l ;  
1508 1626 2 WHILE .a[nwl-1] EQL %c ' '  
1509 1627 2 DO  
1510 1628 2 if (nwl = .nwl-1) leq 0 then return 0;  
1511 1629 2 ! skip over possible leading spaces  
1512 1630 2 WHILE .a[i] eql %c '  
1513 1631 2 DO  
1514 1632 2 BEGIN  
1515 1633 2 INCR c from 0 to .nwl-1  
1516 1634 3 DO
```

```
1517      1635      if (.a[c] = .a[c+1];  
1518      1636      END;  
1519      1637      nd[dsc$w_length] = .nd[dsc$w_length] - (.l-.nwl);  
1520      1638      IF .a[i] eql %c'+' OR .a[i] eql %c'-'  
1521      1639      THEN BEGIN  
1522      1640          s = 3;  
1523      1641          i = .i + 1;  
1524      1642          nd[dsc$w_dtype] = dsc$w_dtype_nl ;  
1525      1643      END  
1526      1644      ELSE nd[dsc$w_dtype] = dsc$w_dtype_nro ;  
1527      1645      INCR c from .i to .nwl-1  
1528      1646      DO BEGIN  
1529      1647          if .a[c] eql %c'.'  
1530      1648      then if .dp  
1531      1649      then return 0  
1532      1650      else begin  
1533      1651          dp = 1;  
1534      1652          dpp = .c;  
1535      1653      end  
1536      1654      else if .a[c] eql %c'd' or .a[c] eql %c'D' or  
1537      1655      .a[c] eql %c'e' or .a[c] eql %c'E'  
1538      1656      then if .ep neq 0  
1539      1657      then return 0  
1540      1658      else ep = .c  
1541      1659      else if .a[c] eql %c'+' or .a[c] eql %c'-'  
1542      1660      then if .esn neq 0 or .ep eq 0  
1543      1661      then return 0  
1544      1662      else esn = (if .a[c] eql %c'+' then 1 else -1)  
1545      1663      else if not (.a[c] geq %c'0' and .a[c] leq %c'9')  
1546      1664      then return 0  
1547      1665      END;  
1548      1666      : now construct scale factor for desc and redo the length  
1549      1667  
1550      1668  
1551      1669  
1552      1670  
1553      1671  
1554      1672  
1555      1673  
1556      1674  
1557      1675  
1558      1676  
1559      1677  
1560      1678  
1561      1679  
1562      1680  
1563      1681  
1564      1682  
1565      1683  
1566      1684  
1567      1685  
1568      1686  
1569      1687      if .ep neq 0  
1570      1688  
1571      1689  
1572      1690  
1573      1691 2      then
```

```

1574      1692      begin
1575      1693      i = .nwl-.ep-1;
1576      1694      if .esn eql 0
1577      1695      then
1578      1696      begin
1579      1697      a[.ep] = %c'+';
1580      1698      cvtsp(i,a[.ep],max_packed_size,p[0]);
1581      1699      end
1582      1700      else
1583      1701      begin
1584      1702      i = .i -1 ;
1585      1703      cvtsp(i,a[.ep+1],max_packed_size,p[0]);
1586      1704      end;
1587      1705      cvtpl(max_packed_size,p[0],ln);
1588      1706      nd[dsc$b_scale] = .ln ;
1589      1707      nd[dsc$w_length] = .ep ;
1590      1708      nwl = .ep ;
1591      1709      end;

1592      1710
1593      1711
1594      1712      if .dp eql 0
1595      1713      then
1596      1714      0
1597      1715      else
1598      1716      begin
1599      1717      ln = (.nwl-.dpp-1);
1600      1718      nd[dsc$b_scale] = .nd[dsc$b_scale] - .ln ;
1601      1719      nd[dsc$w_length] = .nd[dsc$w_length] - 1 ;
1602      1720      ch$move(.ln,a[.dpp+1],p[0]);
1603      1721      ch$move(.ln,p[0],a[.dpp]);
1604      1722      end;
1605      1723      if .nd[dsc$b_dtype] eql dsc$k_dtype_nl
1606      1724      then
1607      1725      nd[dsc$w_length] = .nd[dsc$w_length] - 1;

1608      1726
1609      1727
1610      1728      return .s                            ! catch all return
1611      1729 1 END:                                 ! End of digit_scan

```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

0000001F 0018C P.AAY: .LONG 31

MAX\_PACKED\_SIZE= P.AAY

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

		OFFC 00000
5E	01	08 2C C2 00002
	08	AC D1 00005
45	8F	04 DE 12 00009
		BC 91 0000B
		3E 13 00010

.ENTRY	DBG\$DIGIT_SCAN, Save R2,R3,R4,R5,R6,R7,R8,-	: 1578
SUBL2	#44, SP	
CMPL	L #1	
BNEQ	1\$	
CMPB	#A, #69	
BEQL	6\$	

1616

65	BF	04	BC	91	00012		CMPB	2A, #101		
	5E		57	13	00017		BEQL	6S	1621	
			01	D0	00019	18:	MOVL	#1, S	1622	
			53	7C	0001C		CLRQ	ESN		
			59	7C	0001E		CLRQ	EP		
			5B	D4	00020		CLRL	DP		
			50	D4	00022		CLRL	F		
		55	08	AC	D0	00024	MOVL	L, NWL	1625	
		57	04	AC	D0	00028	MOVL	A, R7	1626	
		20	FF	A547	91	0002C	CMPB	-1(NWL)[R7], #32		
			05	12	00031		BNEQ	3S		
		F6	55	F5	00033		S0BGTR	NWL, 28	1628	
			18	11	00036		BRB	6S		
		20	6447	91	00038	38:	CMPB	(I)[R7], #32	1630	
			15	12	0003C		BNEQ	7S		
		50	01	CE	0003E		MNEGL	#1, C	1633	
			06	11	00041		BRB	5S		
		F6	6047	01	A047	90	00043	1(C)[R7], (C)[R7]	1635	
			50	55	F2	00049	48:	A0BLSS	NWL, C 48	
		E8	55	F5	0004D		S0BGTR	NWL, 3S	1636	
			0101	31	00050	68:	BRW	25S		
		50	58	0C	AC	D0	00053	MOVL	ND, R8	1639
			55	08	C3	00057	SUBL3	L, NWL, R0		
		68	50	A0	0005C		ADDW2	R0, (R8)		
		28	6447	91	0005F		CMPB	(I)[R7], #43	1641	
			06	13	00063		BEQL	8S		
		2D	6447	91	00065		CMPB	(I)[R7], #45		
			0B	12	00069		BNEQ	9S		
		6E	03	D0	0006B	88:	MOVL	#3, S	1644	
			54	D6	0006E		INCL	I	1645	
	02	AB	10	90	00070		MOV8	#16, 2(R8)	1646	
	02	A8	04	11	00074		BRB	10S	1641	
	02	A8	13	90	00076	98:	MOV8	#19, 2(R8)	1649	
		50	FF	A4	9E	0007A	108:	MOVAB	-1(R4), C	1651
		51	64	11	0007E		BRB	19S		
		2E	6047	9A	00080	118:	MOVZBL	(C)[R7], R1	1654	
			51	91	00084		CMPB	R1, #46		
		C4	5B	E8	00089		BNEQ	12S		
		5B	01	D0	0008C		BLBS	DP, 6S	1656	
		5A	50	D0	0008F		MOVL	#1, DP	1661	
			50	11	00092		MOVL	C, DPP	1662	
	64	BF	51	91	00094	128:	BRB	19S	1656	
			12	13	00098		CMPB	R1, #100	1665	
	44	BF	51	91	0009A		BEQL	13S		
			0C	13	0009E		CMPB	R1, #68		
	65	BF	51	91	000A0		BEQL	13S	1666	
			06	13	000A4		CMPB	R1, #101		
	45	BF	51	91	000A6		BEQL	13S		
			09	12	000AA		CMPB	R1, #69		
		59	59	D5	000AC	138:	BNEQ	14S		
			A0	12	000AE		TSTL	EP	1668	
		50	D0	000B0		BNEQ	6S			
		2F	11	000B3		MOVL	C, EP	1672		
		52	D4	000B5		BRB	19S	1668		
		28	51	91	000B7	148:	CLRL	R2	1674	
			04	12	000BA		CMPB	R1, #43		
							BNEQ	15S		

			52	D6 000BC		INCL	R2			
			05	11 000BE		BRB	16\$			
			51	91 000C0	15\$:	CMPB	R1	#45		
			15	12 000C3		BNEQ	18\$			
			53	D5 000C5	16\$:	TSTL	ESN		1676	
			87	12 000C7		BNEQ	6\$			
			59	D5 000C9		TSTL	EP			
			83	13 000CB		BEQL	6\$			
			52	E9 000CD		BLBC	R2.	17\$		
			01	D0 000D0		MOVL	#1	ESN		
			0F	11 000D3	17\$:	BRB	19\$			
			53	0A 000D8		MNEG	#1	ESN		
			30	51 000DA	18\$:	BRB	19\$			
			75	1F 000DD		CMPB	R1	#48		
			51	91 000DF		BLSSU	25\$			
			51	91 000E2		CMPB	R1	#57		
			70	1A 000E2		BGTRU	25\$			
			55	F2 000E4	19\$:	A0BLSS	NWL.	C.	11\$	
			59	D5 000E8		TSTL	EP			
			3E	13 000EA		BEQL	22\$			
			59	C3 000EC		SUBL3	EP,	NWL,	R6	
			A6	9F 000FO		MOVAB	-1(R6),	I		
			53	D5 000F4		TSTL	ESN			
			11	12 000F6		BNEQ	20\$			
			28	90 000F8		MOVB	#43,	(EP)[R7]		
			6947	6947		CVTSP	I,	(EP)[R7],	MAX_PACKED_SIZE,	P
			54	09 000FC		BRB	21\$			
			0E	11 00107		DECL	I,			
			54	D7 00109	20\$:	CVTSP	I,	1(EP)[R7],	MAX_PACKED_SIZE,	P
			54	09 0010B		CVTPL	MÁX_PACKED_SIZE,	P,	LN	
			EF	36 00117	21\$:	MOVB	LN,	8(R8)		
			08	A8 00000000'		MOVW	EP,	(R8)		
			68	59 00120		MOVL	EP,	NWL		
			55	B0 00124		TSTL	DP			
			59	D0 00127		BEQL	23\$			
			58	D5 0012A	22\$:	SUBL2	DPP,	R5		
			1A	13 0012C		MOVAB	-1(R5),	LN		
			55	5A C2 0012E		SUBB2	LN,	B(R8)		
			56	A5 9E 00131		DECW	(R8)			
			FF	56 82 00135		MOVC3	LN,	1(DPP)[R7],	P	
			08	A8 68 B7 00139		MOVC3	LN,	P,	(DPP)[R7]	
			6A47	01 AA47		CMPB	2(R8),	#16		
			04	AE 56 28 0013B		BNEQ	24\$			
			10	02 A8 28 00142	23\$:	DECW	(R8)			
			50	68 02 12 0014C		MOVL	S,	RO		
			6E	B7 0014E	24\$:	RET				
			04	D0 00150		CLRL	RO			
			50	04 00153		RET				
			04	D4 00154	25\$:					
			04	00156						

; Routine Size: 343 bytes. Routine Base: DBG\$CDDE + 0833

; 1612 1730 1



```

: 1671    1788  2      THEN
: 1672    1789  3      BEGIN
: 1673    1790  3
: 1674    1791  3      ! Input_addr definitely corresponds to some address in the context area.
: 1675    1792  3      ! Map it to the user runframe.
: 1676    1793  3      .output_addr = runframe_addrs_vect [.input_addr - dbg$reg_values [0]];
: 1677    1794  3
: 1678    1795  3      RETURN sts$k_success;
: 1679    1796  3
: 1680    1797  3      END
: 1681    1798  2      ELSE
: 1682    1799  3      BEGIN
: 1683    1800  3      ! No match
: 1684    1801  3      RETURN sts$k_severe;
: 1685    1802  3
: 1686    1803  3      END;
: 1687    1804  2
: 1688    1805  1      END;           ! End of dbg$map_to_reg_addr

```

				0004 00000	.ENTRY	DBG\$MAP TO REG_ADDR, Save R2	: 1731
				52 00000000G 00 9E 00002	MOVAB	DBG\$REG_VALUES, R2	: 1781
				51 00000000G 00 9E 00009	MOVAB	DBG\$RUNFRAME+4, RUNFRAME_ADDRS_VECT	: 1785
				50 62 9E 00010	MOVAB	DBG\$REG_VALUES, R0	
				50 04 AC D1 00013	CMPL	INPUT_ADDR, R0	
				18 1F 00017	BLSSU	1\$	
				50 44 A2 9E 00019	MOVAB	DBG\$REG_VALUES+68, R0	: 1787
				50 04 AC D1 0001D	CMPL	INPUT_ADDR, R0	
				11 1E 00021	BGEQU	1\$	
				50 62 9E 00023	MOVAB	DBG\$REG_VALUES, R0	: 1794
	08 50	04 BC		50 C3 00026	SUBL3	R0, INPUT_ADDR, R0	
				51 C1 0002B	ADDL3	RUNFRAME_ADDRS_VECT, R0, BOUTPUT_ADDR	: 1799
				01 D0 00030	MOVL	#1, R0	
				04 00033	RET		
				04 D0 00034 1\$:	MOVL	#4, R0	: 1803
				04 00037	RET		: 1805

; Routine Size: 56 bytes, Routine Base: DBG\$CODE + 098A

; 1689 1806 1

```
1691 1807 1 GLOBAL ROUTINE DBG$EXACT_MAP_TO_REG (INPUT_ADDR, REG_ADDR) =
1692 1808 1
1693 1809 1
1694 1810 1 ++ FUNCTIONAL DESCRIPTION:
1695 1811 1
1696 1812 1 This routine checks to see if the input address can be mapped to the
1697 1813 1 exact starting address of one of the context register value save areas.
1698 1814 1 If it can, then the address is mapped to the starting address of the
1699 1815 1 corresponding runframe registers.
1700 1816 1
1701 1817 1 FORMAL PARAMETERS:
1702 1818 1
1703 1819 1 INPUT_ADDR - A longword containing the address to be mapped
1704 1820 1
1705 1821 1 REG_ADDR - The address of a longword to contain the address of
1706 1822 1 the mapped-to register
1707 1823 1
1708 1824 1 IMPLICIT INPUTS:
1709 1825 1
1710 1826 1 dbg$runframe [dbg$1_user_regs] - the beginning address of the runframe
1711 1827 1 registers
1712 1828 1
1713 1829 1 IMPLICIT OUTPUTS:
1714 1830 1
1715 1831 1 NONE
1716 1832 1
1717 1833 1 ROUTINE VALUE:
1718 1834 1 An unsigned integer longword completion code
1719 1835 1
1720 1836 1 COMPLETION CODES:
1721 1837 1
1722 1838 1 STSSK_SUCCESS (1) - Success. Input address mapped to register address.
1723 1839 1
1724 1840 1 STSSK_SEVERE (4) - Failure. Input address not mapped.
1725 1841 1
1726 1842 1
1727 1843 1 SIDE EFFECTS:
1728 1844 1
1729 1845 1 NONE
1730 1846 1
1731 1847 1 -- BEGIN
1732 1848 2
1733 1849 2
1734 1850 2 LOCAL RUNFRAME_ADDRESS: ! Address within runframe
1735 1851 2 ! area
1736 1852 2
1737 1853 2
1738 1854 2 ! See if the input address maps to any place in the runframe regs
1739 1855 2
1740 1856 2 IF dbg$map_to_reg_addr (.input_addr, runframe_address)
1741 1857 2 THEN BEGIN
1742 1858 2
1743 1859 2 ! See if the resulting mapped address corresponds exactly to a reg
1744 1860 2 ! beginning address
1745 1861 2
1746 1862 2
1747 1863 2 IF ((.runframe_address - dbg$runframe [dbg$1_user_regs]) MOD 4) EQL 0
```

```

: 1748    1864  5      THEN
: 1749    1865  4      BEGIN
: 1750    1866  4
: 1751    1867  4      ! Exact match to runframe reg
: 1752    1868  4
: 1753    1869  4      .reg_addr = .runframe_address;
: 1754    1870  4      RETURN sts$k_success;
: 1755    1871  3      END;
: 1756    1872  2
: 1757    1873  2
: 1758    1874  2      ! No match
: 1759    1875  2
: 1760    1876  2      RETURN sts$k_severe;
: 1761    1877  2
: 1762    1878  1      END;          ! End of dbg$exact_map_to_reg

```

					.ENTRY	DBG\$EXACT_MAP_TO_REG, Save nothing	: 1807
					SUBL2	#4, SP	
					PUSHL	SP	: 1856
					INPUT ADDR		
					CALLS	#2, DBG\$MAP_TO_REG_ADDR	
					BLBC	R0, 1\$	
					MOVAB	DBG\$RUNFRAME+4, R0	: 1863
					SUBL3	R0, RUNFRAME_ADDRESS, R0	
					EMUL	#1, R0, #0, -(SP)	
					EDIV	#4, (SP)+, R0, R0	
					TSTL	R0	
					BNEQ	1\$	
					MOVL	RUNFRAME_ADDRESS, @REG_ADDR	: 1869
					MOVL	#1, R0	: 1870
					RET		
					MOVL	#4, R0	: 1876
					RET		: 1878

; Routine Size: 54 bytes, Routine Base: DBG\$CODE + 09C2

```
1764      1879 1 MACROS:  
1765      1880 1  
1766      1881 1  
1767      1882 1 The keyword_table is made of four-tuple entries.  
1768      1883 1 1) the language index (0 - n),  
1769      1884 1 2) the number of characters in the minimal abbreviation,  
1770      1885 1 3) the number of characters in the language name,  
1771      1886 1 4) the language name as an ASCII string.  
1772      1887 1  
1773      1888 1 Macro KEY_NAME formats table entries for the language name table.  
1774      1889 1 Each entry has three formals:  
1775      1890 1 1) the ASCII string representing a language name,  
1776      1891 1 2) the length of that ASCII string abbreviated,  
1777      1892 1 3) the language index for that language  
1778      1893 1  
1779      1894 1 MACRO  
1780      1895 1 M KEY_NAME (KNAME, KABBREV, KEQUIV) =  
1781          KEQUIV, KABBREV, %CHARCOUNT (KNAME), %ASCII KNAME%;  
1782      1896 1  
1783      1897 1  
1784      1898 1 BIND  
1785      1899 1 LANGUAGE_TABLE = UPLIT BYTE(  
1786      1900 1  
1787      1901 1     KEY_NAME ('MACRO', 2. DBGSK_MACRO),  
1788      1902 1     KEY_NAME ('FORTRAN', 2. DBGSK_FORTRAN),  
1789      1903 1     KEY_NAME ('BLISS', 2. DBGSK_BLISS),  
1790      1904 1     KEY_NAME ('COBOL', 2. DBGSK_COBOL),  
1791      1905 1     KEY_NAME ('BASIC', 2. DBGSK_BASIC),  
1792      1906 1     KEY_NAME ('PLI', 2. DBGSK_PLI),  
1793      1907 1     KEY_NAME ('PASCAL', 2. DBGSK_PASCAL),  
1794      1908 1     KEY_NAME ('C', 1. DBGSK_C),  
1795      1909 1     KEY_NAME ('RPG', 2. DBGSK_RPG),  
1796      1910 1     KEY_NAME ('ADA', 2. DBGSK_ADA),  
1797      1911 1     KEY_NAME ('UNKNOWN', 3. DBGSK_UNKNOWN),  
1798      1912 1  
        1913 1     0 ) : VECTOR [, BYTE];
```

```

1800 1914 1 GLOBAL ROUTINE DBGSSET_LANG (LANG_STR_DESC, LANGUAGE_MODULE) =
1801 1915 1
1802 1916 1 FUNCTION
1803 1917 1 This routine loads the pointers to the current parsing tables
1804 1918 1 with those of the new language.
1805 1919 1
1806 1920 1 INPUTS
1807 1921 1 LANG_STR_DESC - no longer used (always 0). Eventually, this
1808 1922 1 parameter should be eliminated (requires
1809 1923 1 changing the 4 places this routine is called.)
1810 1924 1 LANGUAGE_MODULE - If "lang_str_desc" is zero this parameter holds the
1811 1925 1 language number as defined in DBGLIB.REQ.
1812 1926 1
1813 1927 1 OUTPUTS
1814 1928 1 The language index of the language to which DEBUG will be set is
1815 1929 1 returned as the routine value.
1816 1930 1
1817 1931 1 BEGIN
1818 1932 1 LOCAL
1819 1933 1 DEF_RADIX;
1820 1934 1
1821 1935 1 ; Change language setting
1822 1936 1
1823 1937 1
1824 1938 1 DBGS$GB_LANGUAGE = .LANGUAGE_MODULE;
1825 1939 1 DBGPARSER SET LANGUAGE (.DBGS$GB_LANGUAGE);
1826 1940 1 DBGS$NCHANGE TO NEW ();
1827 1941 1 DBGS$SET_MOD_DEF ();
1828 1942 1 DBGS$SET_STP_DEF ();
1829 1943 1 DBGS$SET_SEARCH_DEF ();
1830 1944 1
1831 1945 1 ; Set up the default radix settings for this language.
1832 1946 1
1833 1947 1 def radix = dbgs$nget_trans_radix(dbgs$k_default);
1834 1948 1 dbgs$gb_radix[dbgs$b_radix_input] = .def_radix;
1835 1949 1 dbgs$gb_radix[dbgs$b_radix_output] = .def_radix;
1836 1950 1 dbgs$gb_radix[dbgs$b_radix_output_over] = dbgs$k_default;
1837 1951 1
1838 1952 1 RETURN .LANGUAGE_MODULE;
1839 1953 1 END;

```

## .PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

							P.AAZ:	.BYTE	0, 2, 5	
		4F	52	05	02	00	00190	.ASCII	\MACRO\	
				43	41	40	00193	.BYTE	1, 2, 7	
		4E	41	07	02	01	00198	.ASCII	\FORTRAN\	
			52	54	52	4F	001A2	.BYTE	2, 2, 5	
				53	53	49	4C	001A5	.ASCII	\BLISS\
				53	53	05	02	001AA	.BYTE	3, 2, 5
			4C	4F	42	4F	001AD	.ASCII	\COBOL\	
				43	49	53	41	001B2	.BYTE	4, 2, 5
					03	02	04	001B5	.ASCII	\BASIC\
					03	02	05	001BA	.BYTE	5, 2, 3
					49	4C	50	001BD	.ASCII	\PLI\

4C	41	43	06	02	06	001C0	.BYTE	6	2	6
			53	41	50	001C3	.ASCII	\PASCAL\		
			01	01	07	001C9	.BYTE	7	1.	1
					43	001CC	.ASCII	\C\		
				03	02	08	.BYTE	8	2	3
				47	50	52	.ASCII	\RPG\		
				03	02	09	.BYTE	9	2	3
				41	44	61	.ASCII	\ADA\		
				07	03	0A	.BYTE	10,	3,	7
			4E	57	4F	4E	.ASCII	\UNKNOWN\		
				4B	4E	55	.BYTE	0		
					00	001E5	.BYTE			

LANGUAGE\_TABLE= P.AAZ

							.PSECT	DBGSCODE, NOWRT, SHR, PIC, O	
							.ENTRY	DBGSET LANG, Save R2,R3	: 1914
							MOVAB	DBGSGB_LANGUAGE, R3	
							MOVAB	DBGSGB_RADIX, R2	
							MOVVB	LANGUAGE MODULE, DBGSGB_LANGUAGE	
							MOVZBL	DBGSGB_LANGUAGE, -(SP)	
							CALLS	#1, DBGSParser_Set_Language	
							CALLS	#0, DBGSNChange_To_New	
							CALLS	#0, DBGSSET_MOD_DEF	
							CALLS	#0, DBGSSET_STP_DEF	
							CALLS	#0, DBGSSET_SEARCH_DEF	
							PUSHL	#1	
							CALLS	#1, DBGSNGET_TRANS_RADIX	
							MOVFB	DEF_RADIX, DBGSGB_RADIX	
							MOVFB	DEF_RADIX, DBGSGB_RADIX+1	
							MOVL	#1, DBGSGB_RADIX+2	
							RET	LANGUAGE_MODULE, R0	

: Routine Size: 83 bytes.   Routine Base: DBGSCODE + 09F8

```
1841 1954 1 GLOBAL ROUTINE DBG$LANGUAGE (LANG_ENCODING) =  
1842 1955 1  
1843 1956 1 FUNCTION  
1844 1957 1 Return a pointer to a counted string which is  
1845 1958 1 the name of the given language.  
1846 1959 1 This function exists simply to consolidate this naming  
1847 1960 1 translation into one place.  
1848 1961 1  
1849 1962 1  
1850 1963 1  
1851 1964 1  
1852 1965 1  
1853 1966 1  
1854 1967 1  
1855 1968 1  
1856 1969 1  
1857 1970 1  
1858 1971 1  
1859 1972 1  
1860 1973 1  
1861 1974 2  
1862 1975 2  
1863 1976 2  
1864 1977 2  
1865 1978 2  
1866 1979 2  
1867 1980 2  
1868 1981 2  
1869 1982 2  
1870 1983 2  
1871 1984 2  
1872 1985 2  
1873 1986 2  
1874 1987 2  
1875 1988 2  
1876 1989 2  
1877 1990 2  
1878 1991 2  
1879 1992 2  
1880 1993 2  
1881 1994 2  
1882 1995 2  
1883 1996 2  
1884 1997 2  
1885 1998 2  
1886 1999 2  
1887 2000 2  
1888 2001 2  
1889 2002 2  
1890 2003 2  
1891 2004 2  
1892 2005 2  
1893 2006 2  
1894 2007 2  
1895 2008 2  
1896 2009 2  
1897 2010 2  
1  
FUNCTION  
Return a pointer to a counted string which is  
the name of the given language.  
This function exists simply to consolidate this naming  
translation into one place.  
INPUTS  
LANG_ENCODING - The numeric encoding used internally to  
represent the language. This is the same  
value that comes in the DST MODULE records for  
each language, and it is the same value that we  
store in DBGSGL_LANGUAGE.  
OUTPUTS  
A pointer to a counted string which names the indicated language  
is returned as the routine value.  
BEGIN  
Just return the desired pointer.  
CASE .LANG_ENCODING FROM DBGSK_MACRO TO DBGSK_UNKNOWN OF  
SET  
[DBGSK_MACRO]:  
    RETURN UPLIT BYTE(%ASCIC 'MACRO');  
[DBGSK_FORTRAN]:  
    RETURN UPLIT BYTE(%ASCIC 'FORTRAN');  
[DBGSK_BLISS]:  
    RETURN UPLIT BYTE(%ASCIC 'BLISS');  
[DBGSK_COBOL]:  
    RETURN UPLIT BYTE(%ASCIC 'COBOL');  
[DBGSK_BASIC]:  
    RETURN UPLIT BYTE(%ASCIC 'BASIC');  
[DBGSK_PLI]:  
    RETURN UPLIT BYTE(%ASCIC 'PLI');  
[DBGSK_PASCAL]:  
    RETURN UPLIT BYTE(%ASCIC 'PASCAL');  
[DBGSK_C]:  
    RETURN UPLIT BYTE(%ASCIC 'C');  
[DBGSK_RPG]:  
    RETURN UPLIT BYTE(%ASCIC 'RPG');  
[DBGSK_ADA]:  
    RETURN UPLIT BYTE(%ASCIC 'ADA');
```

1898  
1899  
1900  
1901  
1902  
1903  
1904

2011 2012 2013 2014 2015 2016 2017

[INRANGE, OUTRANGE]:  
RETURN UPLT BYTE(%ASCIC 'UNKNOWN');

TES:

END:

## .PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

4E	41	4F	52	43	41	4D	05	001E4	P.ABA:	.ASCII	<5>\MACRO\		
		52	54	52	4F	46	07	001EA	P.ABB:	.ASCII	<7>\FORTRAN\		
		55	53	49	4C	42	05	001F2	P.ABC:	.ASCII	<5>\BLISS\		
		4C	4F	42	4F	43	05	001F8	P.ABD:	.ASCII	<5>\COBOL\		
		43	49	53	41	42	05	001FE	P.ABE:	.ASCII	<5>\BASIC\		
				49	4C	50	03	00204	P.ABF:	.ASCII	<3>\PLI\		
		4C	41	43	53	41	50	06	00208	P.ABG:	.ASCII	<6>\PASCAL\	
					43	01		0020F	P.ABH:	.ASCII	<1>\C\		
					47	50	52	03	00211	P.ABI:	.ASCII	<3>\RPG\	
					41	44	41	03	00215	P.ABJ:	.ASCII	<3>\ADA\	
		4E	57	4F	4E	4B	4E	55	07	00219	P.ABK:	.ASCII	<7>\UNKNOWN\

## .PSECT DBG\$CODE,NOWRT, SHR, PIC,0

0029	0A	52	00000000'	EF	0004	00000			.ENTRY	DBG\$LANGUAGE, Save R2	1954
0030	0024	00	04	AC	9E	00002			MOVAB	P.ABK, R2	2010
	0038	001F		CF	00009				CASEL	LANG ENCODING, #0, #10	
	0016	0033		001A	0000E		1\$:		.WORD	3\$-1\$,-	
		0047		002E	00016					4\$-1\$,-	
				0042	0001E					5\$-1\$,-	
										6\$-1\$,-	
										7\$-1\$,-	
										8\$-1\$,-	
										9\$-1\$,-	
										10\$-1\$,-	
										11\$-1\$,-	
										12\$-1\$,-	
										2\$-1\$	
		50		62	9E	00024	2\$:	MOVAB	P.ABK, R0	2013	
					04	00027		RET		1983	
		50	CB	A2	9E	00028	3\$:	MOVAB	P.ABA, R0	2010	
					04	0002C		RET			
		50	D1	A2	9E	0002D	4\$:	MOVAB	P.ABB, R0	1986	
					04	00031		RET			
		50	D9	A2	9E	00032	5\$:	MOVAB	P.ABC, R0	1989	
					04	00036		RET			
		50	DF	A2	9E	00037	6\$:	MOVAB	P.ABD, R0	2010	
					04	0003B		RET			
		50	E5	A2	9E	0003C	7\$:	MOVAB	P.ABE, R0	1992	
					04	00040		RET			
		50	EB	A2	9E	00041	8\$:	MOVAB	P.ABF, R0	2010	
					04	00045		RET			
		50	EF	A2	9E	00046	9\$:	MOVAB	P.ABG, R0	1995	

50	F6	A2	04 0004A	RET
			9E 0004B	MOVAB P.ABH, R0
50	F8	A2	04 0004F	RET
			9E 00050	MOVAB P.ABI, R0
50	FC	A2	04 00054	RET
			9E 00055	MOVAB P.ABJ, R0
			04 00059	RET

: 2010  
: 2004  
: 2010  
: 2007  
: 2010  
: 2017

: Routine Size: 90 bytes, Routine Base: DBGSCODE + 0A4B

```
1905      2018    1 BIND
1906      2019    1
1907      2020    1
1908      2021    1 deficf_name     = UPLIT BYTE(%ASCII 'DEBUG.COM'),
1909      2022    1 deficf_size      = ICHARCOUNT(%ASCII 'DEBUG.COM');
1910      2023    1 MACRO
1911      2024    1
1912      2025    1
1913      2026    1
1914      2027    1
1915      2028    1
1916      2029    1
1917      2030    1
1918      2031    1
1919      2032    1
1920      2033    1
1921      2034    1
1922      2035    1
1923      2036    1
1924      2037    1
1925      2038    1
1926      2039    1
1927      2040    1
1928      2041    1
1929      2042    1
                                icf_message (prefix) =
                                BEGIN
                                BIND
                                enter_phrase = UPLIT BYTE(8, %ASCII 'entering').
                                exit_phrase = UPLIT BYTE(7, %ASCII 'exiting');
                                LOCAL
                                phrase;
                                IF prefix EQ 1
                                THEN
                                phrase = enter_phrase
                                ELSE
                                phrase = exit_phrase;
                                SIGNAL (dbgS_verifyicf, 3, .phrase, .fab_ptr[fab$B_fns], .fab_ptr[fab$L_fns]); ! Info message
                                END % ;
```

```
1931 2043 1 GLOBAL ROUTINE DBGSCIS_CONNECTICF (SIGNAL_FLAG) : NOVALUE =
1932 2044 1
1933 2045 1 ++
1934 2046 1 FUNCTIONAL DESCRIPTION:
1935 2047 1
1936 2048 1
1937 2049 1 FORMAL PARAMETERS:
1938 2050 1 SIGNAL_FLAG - TRUE if called from normal command processing and
1939 2051 1 we should signal warning message on failure.
1940 2052 1 FALSE if called from setting up DEBUG initialization
1941 2053 1 file. In this case just signal informational.
1942 2054 1
1943 2055 1 IMPLICIT INPUTS:
1944 2056 1 NONE
1945 2057 1
1946 2058 1 IMPLICIT OUTPUTS:
1947 2059 1 NONE
1948 2060 1
1949 2061 1 ROUTINE VALUE:
1950 2062 1 An unsigned integer longword completion code
1951 2063 1
1952 2064 1 COMPLETION CODES:
1953 2065 1 NONE
1954 2066 1
1955 2067 1 SIDE EFFECTS:
1956 2068 1 NONE
1957 2069 1
1958 2070 1
1959 2071 1
1960 2072 1
1961 2073 1
1962 2074 1
1963 2075 1 --
1964 2076 2 BEGIN
1965 2077 2
1966 2078 2 LOCAL
1967 2079 2 dummy_mess_vect,
1968 2080 2 status, ! Return status
1969 2081 2 fab_ptr : REF $FAB_DECL, ! ptr to allocated FAB storage
1970 2082 2 rab_ptr : REF $RAB_DECL, ! ptr to allocated RAB storage
1971 2083 2 ind_com_filesp : REF VECTOR [,BYTE]; ! Filespec counted string
1972 2084 2
1973 2085 2 ind_com_filesp = .dbg$gl_ind_com_file;
1974 2086 2
1975 2087 2 ! Allocate FAB and RAB storage
1976 2088 2
1977 2089 2 fab_ptr = dbg$get_memory ((fab$e_bln + 3)/ XUPVAL);
1978 2090 2 rab_ptr = dbg$get_memory ((rab$e_bln + 3)/ XUPVAL);
1979 2091 2
1980 2092 2 ! Initialize the FAB and the RAB
1981 2093 2
1982 P 2094 2 SFAB_INIT (FAB=.fab_ptr, FAC=GET, FNA=.ind_com_filesp + 1, FNS=.ind_com_filesp[0],
1983 2095 2 DNA=defict_name, DNS=defict_size);
1984 2096 2 SRAB_INIT (RAB=.rab_ptr, FAB=.fab_ptr);
1985 2097 2
1986 2098 2 ! Put them on the command input stream
1987 2099 2
```

```
1988      2100 2
1989      2101 2
1990      2102 2
1991      2103 2
1992      2104 2
1993      2105 2
1994      2106 2
1995      2107 2
1996      2108 2
1997      2109 2
1998      2110 2
1999      2111 2
2000      2112 2
2001      2113 2
2002      2114 2
2003      2115 2
2004      2116 2
2005      2117 2
2006      2118 2
2007      2119 2
2008      2120 2
2009      2121 2
2010      2122 2
2011      2123 2
2012      2124 2
2013      2125 2
2014      2126 2
2015      2127 2
2016      2128 2
2017      2129 2
2018      2130 2
2019      2131 2
2020      2132 2
2021      2133 2
2022      2134 2
2023      2135 2
2024      2136 2
2025      2137 2
2026      2138 2
2027      2139 2
2028      2140 2
2029      2141 2
2030      2142 2
2031      2143 2
2032      2144 2
2033      2145 2
2034      2146 2
2035      2147 2
2036      2148 2
2037      2149 2
2038      2150 2
2039      2151 2
2040      2152 2
2041      2153 2
2042      2154 2
2043      2155 2
2044      2156 2

      dbgScis_add (.rab_ptr, 0, cis_rab, 0, 0);
      ! Set up the local define list for the command procedure.
      IF NOT dbg$def_pr_entry (dummy_mess_vect)
      THEN
          ! Signal the error.
          BEGIN
              EXTERNAL ROUTINE
              lib$signal: ADDRESSING_MODE(GENERAL);
              BUILTIN
              CALLG;
              CALLG (.dummy_mess_vect, lib$signal);
              END;
      ! Open and connect the file
      status = $OPEN (FAB=.fab_ptr);
      IF NOT .status
      THEN
          BEGIN
              LOCAL
                  msg_desc : BLOCK [8,BYTE];
                  msg_desc[dsc$w_length] = .fab_ptr[fab$bfns];
                  msg_desc[dsc$w_pointer] = .fab_ptr[fab$lfns];
          ! Flag link for removal so we won't try to read from it again
          dbgSgl_cishead[cis$w_rem_flag] = 1;
          IF .signal_flag
          THEN
              SIGNAL (shr$openin + dbg_fac_code, 1, msg_desc,
                      .fab_ptr[fab$ls], .fab_ptr[fab$ls$tv])
          ELSE
              BEGIN
                  SIGNAL (dbg$unaopnini, 1, msg_desc,
                          .fab_ptr[fab$ls], .fab_ptr[fab$ls$tv]);
                  RETURN;
              END;
          END;
      ! Connect the RAB to the just opened FAB
      status = $CONNECT (RAB=.rab_ptr);
      IF NOT .status
      THEN
          BEGIN
              LOCAL
                  msg_desc : BLOCK [8,BYTE];

```

```

2045      2157      msg_desc[dsc$w_length] = .fab_ptr[fab$bfns];
2046      2158      msg_desc[dsc$w_pointer] = .fab_ptr[fab$bfna];
2047      2159
2048      2160
2049      2161      ! Flag link for removal so we won't try to read from it again
2050      2162      dbg$gl_cishead[cis$v_rem_flag] = 1;
2051      2163
2052      2164      IF .signal_flag
2053      2165      THEN
2054      2166          SIGNAL (shr$openin + dbg_fac_code, 1, msg_desc,
2055      2167              .fab_ptr[fab$sts], .fab_ptr[fab$stv])
2056      2168      ELSE
2057      2169          SIGNAL (dbg$unaopnini, 1, msg_desc,
2058      2170              .fab_ptr[fab$sts], .fab_ptr[fab$stv]);
2059      2171
2060      2172
2061      2173
2062      2174      END;
2063      2175
2064      2176      IF .dbg$gb_def_out [out_verify]
2065      2177      THEN
2066      2178          icf_message(1);
2067      2179
2068      2180      RETURN;
2069      2181
2070      2182 1 END:      ! End of dbg$cis_connecticf

```

										.PSECT	DBG\$PLIT,NOWRT, SHR, PIC,0	
4D	4F	43	2E	47	55	42	45	44	00221	P.ABL:	.ASCII	\DEBUG.COM\
67	6E	69	72	65	74	6E	65	08	0022A	P.ABM:	.BYTE	8
								07	0022B	P.ABN:	.ASCII	\entering\
								07	00233	P.ABN:	.BYTE	7
								65	00234	P.ABN:	.ASCII	\exiting\
										DEFICF_NAME=	P.ABL	
										DEFICF_SIZE=	9	
										ENTER_PHRASE=	P.ABM	
										EXIT_PHRASE=	P.ABN	
										.EXTRN	LIB\$SIGNAL	
										.PSECT	DBG\$CODE,NOWRT, SHR, PIC,0	
OFFC 00000										.ENTRY	DBG\$CIS_CONNECTICF, Save R2,R3,R4,R5,R6,R7,-; 2043	
5B	00000000G	00	9E	00002		MOVAB	R8,R9,R10,R11					
5A	00000000G	00	9E	00009		MOVAB	DBG\$GL_CI\$HEAD, R11					
59	00000000G	00	9E	00010		MOVAB	DBG\$GET_MEMORY, R10					
5E		0C	C2	00017		SUBL2	LIB\$SIGNAL, R9					
58	00000000G	00	DD	0001A		MOVL	#12 SP					
		14	DD	00021		PUSHL	DBG\$GL_IND_COM_FILE, IND_COM_FILESP					
6A		01	FB	00023		CALLS	#20					
56		50	DD	00026		MOVL	#1, DBG\$GET_MEMORY					
		11	DD	00029		PUSHL	RO, FAB_PTR					
6A		01	FB	0002B		CALLS	#1, DBG\$GET_MEMORY					

0050	BF	00	57	50	00 2C 0002E	MOVL	R0, RAB_PTR		2095
			6E	00	2C 00031	MOVCS	#0, (SPT), #0, #80, (FAB_PTR)		
			66	66	00038	MOVW	#20483, (FAB_PTR)		
			16 A6	8F	BO 00039	MOVB	#2, 22(FAB_PTR)		
			1F A6	02	90 0003E	MOVB	#2, 31(FAB_PTR)		
			2C A6	02	90 00042	MOVAB	1(R8), 44(FAB_PTR)		
			30 A6	A8	9E 00046	MOVAB	DEFICF NAME, 48(FAB_PTR)		
			34 A6	EF	9E 0004B	MOVAB	(IND COM FILESP), 52(FAB_PTR)		
			35 A6	68	90 00053	MOVB	#9, 53(FAB_PTR)		
				09	90 00057	MOVB	#0, (SP), #0, #68, (RAB_PTR)		2096
0044	BF	00	6E	00	2C 0005B	MOVCS	#0, (SP), #0, #68, (RAB_PTR)		
				67	66	MOVW	#17409, (RAB_PTR)		
			3C A7	8F	BO 00063	MOVL	FAB_PTR, 60(RAB_PTR)		2100
				56	DO 00068	CLRL	-(SP)		
				7E	7C 0006C	PUSHL	#1		
				01	DD 0006E	CLRL	-(SP)		
				7E	D4 00070	PUSHL	RAB_PTR		
				57	DD 00072	CALLS	#5, -DBGSCIS_ADD		2104
		0000V CF		05	FB 00074	PUSHL	SP		
		00000000G	00	5E	DD 00079	CALLS	#1, DBGSDEF_PR_ENTRY		
		04	01	FB 0007B	BLBS	R0, 1\$			
		69	00	50 E8 00082	CALLG	@DUMMY MESS_VECT, LIB\$SIGNAL			
		00000000G	00	56 DD 00089	18:	PUSHL	FAB_PTR		2114
			01	FB 0008B	CALLS	#1, -SYS\$OPEN		2119	
			52	50 DO 00092	MOVL	R0, STATUS			
			3A	52 E8 00095	BLBS	STATUS, 3\$		2120	
		04 AE	34 A6 9B 00098	MOVZBW	52(FAB_PTR), MSG_DESC			2128	
		08 AE	2C A6 DO 0009D	MOVL	44(FAB_PTR), MSG_DESC+4			2129	
		50	68 DO 000A2	MOVL	DBG\$GL-(CISHHEAD, R0)			2134	
		12 A0	01 88 000A5	BISB2	#1, 18(R0)				
		14	AC E9 000A9	BLBC	SIGNAL FLAG, 2\$			2136	
		7E	08 A6 7D 000AD	MOVQ	8(FAB_PTR), -(SP)			2139	
			0C AE 9F 000B1	PUSHAB	MSG_DESC			2138	
			01 DD 000B4	PUSHL	#1				
			8F DD 000B6	PUSHL	#135320				
		69	00021098	05 FB 000BC	CALLS	#5, LIB\$SIGNAL			
			11 11 000BF	BRB	3\$				
		7E	08 A6 7D 000C1	28:	MOVQ	8(FAB_PTR), -(SP)		2143	
			0C AE 9F 000C5	PUSHAB	MSG_DESC			2142	
			01 DD 000C8	PUSHL	#1				
			8F DD 000CA	PUSHL	#165507				
			66 11 000D0	BRB	7\$				
		00000000G	00	57 DD 200D2	38:	PUSHL	RAB_PTR		2150
			01 FB 000D4	CALLS	#1, -SYS\$CONNECT				
			50 DO 000DB	MOVL	R0, STATUS				
			52 E8 000DE	BLBS	STATUS, 6\$			2151	
			38 A6 9B 000E1	MOVZBW	52(FAB_PTR), MSG_DESC			2157	
		04 AE	34 A6 DO 000E6	MOVL	44(FAB_PTR), MSG_DESC+4			2158	
		08 AE	68 DO 000EB	MOVL	DBG\$GL-(CISHHEAD, R0)			2163	
		50	01 88 000EE	BISB2	#1, 18(R0)				
		12 A0	AC E9 000F2	BLBC	SIGNAL FLAG, 4\$			2165	
		11	08 A6 7D 000F6	MOVQ	8(FAB_PTR), -(SP)			2168	
		7E	0C AE 9F 000FA	PUSHAB	MSG_DESC			2167	
			01 DD 000FD	PUSHL	#1				
			8F DD 000FF	PUSHL	#135320				
			0F 11 00105	BRB	5\$				

7E	08	A6	7D	00107	4\$:	MOVQ	8(FAB PTR), -(SP)	2171
	0C	AE	9F	0010B		PUSHAB	MSG_DESC	2170
		01	DD	0010E		PUSHL	#1	
00028683		8F	DD	00110		PUSHL	#165507	
69		05	FB	00116	5\$:	CALLS	#5, LIB\$SIGNAL	
18 0000000G		00	E9	00119	6\$:	BLBC	DBG\$GB DEF OUT+2, 8\$	2176
50 00000000		EF	9E	00120		MOVAB	ENTER PHRASE, PHRASE	2178
	2C	A6	DD	00127		PUSHL	44(FAB PTR)	
7E	34	A6	9A	0012A		MOVZBL	52(FAB PTR), -(SP)	
		50	DD	0012E		PUSHL	PHRASE	
		03	DD	00130		PUSHL	#3	
00028088		8F	DD	00132		PUSHL	#163979	
69		05	FB	00138	7\$:	CALLS	#5, LIB\$SIGNAL	
				04	0013B 8\$:	RET		2182

; Routine Size: 316 bytes,    Routine Base: DBG\$CODE + 0AA5

```
2072      2183 1 GLOBAL ROUTINE dbg$cis_remove (exit_flag) : NOVALUE =
2073      2184 1 ++
2074      2185 1   FUNCTIONAL DESCRIPTION:
2075      2186 1     Removes the top link from the command input stream and delete the
2076      2187 1     storage for it. If the link has additional dynamic storage related to
2077      2188 1     it, such as a FAB,RAB, input buffer etc., that storage is freed also.
2078      2189 1     Note - this routine now just calls the routine DBGSNCIS_REMOVE in
2079      2190 1     the module DBGNEXCTE.
2080      2191 1
2081      2192 1   FORMAL PARAMETERS:
2082      2193 1
2083      2194 1     exit_flag - TRUE if called from EXIT command.
2084      2195 1
2085      2196 1   IMPLICIT INPUTS:
2086      2197 1     The head of the command input stream
2087      2198 1
2088      2199 1   IMPLICIT OUTPUTS:
2089      2200 1     None
2090      2201 1
2091      2202 1   ROUTINE VALUE:
2092      2203 1     None
2093      2204 1
2094      2205 1   SIDE EFFECTS:
2095      2206 1     The head of the command input stream is reset to what was the
2096      2207 1     "next" link before this routine was called. If SET OUTPUT VERIFY,
2097      2208 1     then a message is generated saying we are exiting the indirect
2098      2209 1     command file.
2099      2210 1
2100      2211 1
2101      2212 2
2102      2213 2
2103      2214 2
2104      2215 2     LOCAL
2105      2216 2     message_vect; ! Dummy message argument vector.
2106      2217 2
2107      2218 2
2108      2219 2
2109      2220 2
2110      2221 2
2111      2222 2
2112      2223 2
2113      2224 2
2114      2225 2
2115      2226 2
2116      2227 2
2117      2228 2
2118      2229 2
2119      2230 2
2120      2231 2
2121      2232 1
2122      2233 1
2123      2234 1
2124      2235 1
2125      2236 1
2126      2237 1
2127      2238 1
2128      2239 1
2129      2240 1
2130      2241 1
2131      2242 1
2132      2243 1
2133      2244 1
2134      2245 1
2135      2246 1
2136      2247 1
2137      2248 1
2138      2249 1
2139      2250 1
2140      2251 1
2141      2252 1
2142      2253 1
2143      2254 1
2144      2255 1
2145      2256 1
2146      2257 1
2147      2258 1
2148      2259 1
2149      2260 1
2150      2261 1
2151      2262 1
2152      2263 1
2153      2264 1
2154      2265 1
2155      2266 1
2156      2267 1
2157      2268 1
2158      2269 1
2159      2270 1
2160      2271 1
2161      2272 1
2162      2273 1
2163      2274 1
2164      2275 1
2165      2276 1
2166      2277 1
2167      2278 1
2168      2279 1
2169      2280 1
2170      2281 1
2171      2282 1
2172      2283 1
2173      2284 1
2174      2285 1
2175      2286 1
2176      2287 1
2177      2288 1
2178      2289 1
2179      2290 1
2180      2291 1
2181      2292 1
2182      2293 1
2183      2294 1
2184      2295 1
2185      2296 1
2186      2297 1
2187      2298 1
2188      2299 1
2189      2300 1
2190      2301 1
2191      2302 1
2192      2303 1
2193      2304 1
2194      2305 1
2195      2306 1
2196      2307 1
2197      2308 1
2198      2309 1
2199      2310 1
2200      2311 1
2201      2312 1
2202      2313 1
2203      2314 1
2204      2315 1
2205      2316 1
2206      2317 1
2207      2318 1
2208      2319 1
2209      2320 1
2210      2321 1
2211      2322 1
2212      2323 1
2213      2324 1
2214      2325 1
2215      2326 1
2216      2327 1
2217      2328 1
2218      2329 1
2219      2330 1
2220      2331 1
2221      2332 1
2222      2333 1
2223      2334 1
2224      2335 1
2225      2336 1
2226      2337 1
2227      2338 1
2228      2339 1
2229      2340 1
2230      2341 1
2231      2342 1
2232      2343 1
2233      2344 1
2234      2345 1
2235      2346 1
2236      2347 1
2237      2348 1
2238      2349 1
2239      2350 1
2240      2351 1
2241      2352 1
2242      2353 1
2243      2354 1
2244      2355 1
2245      2356 1
2246      2357 1
2247      2358 1
2248      2359 1
2249      2360 1
2250      2361 1
2251      2362 1
2252      2363 1
2253      2364 1
2254      2365 1
2255      2366 1
2256      2367 1
2257      2368 1
2258      2369 1
2259      2370 1
2260      2371 1
2261      2372 1
2262      2373 1
2263      2374 1
2264      2375 1
2265      2376 1
2266      2377 1
2267      2378 1
2268      2379 1
2269      2380 1
2270      2381 1
2271      2382 1
2272      2383 1
2273      2384 1
2274      2385 1
2275      2386 1
2276      2387 1
2277      2388 1
2278      2389 1
2279      2390 1
2280      2391 1
2281      2392 1
2282      2393 1
2283      2394 1
2284      2395 1
2285      2396 1
2286      2397 1
2287      2398 1
2288      2399 1
2289      2400 1
2290      2401 1
2291      2402 1
2292      2403 1
2293      2404 1
2294      2405 1
2295      2406 1
2296      2407 1
2297      2408 1
2298      2409 1
2299      2410 1
2300      2411 1
2301      2412 1
2302      2413 1
2303      2414 1
2304      2415 1
2305      2416 1
2306      2417 1
2307      2418 1
2308      2419 1
2309      2420 1
2310      2421 1
2311      2422 1
2312      2423 1
2313      2424 1
2314      2425 1
2315      2426 1
2316      2427 1
2317      2428 1
2318      2429 1
2319      2430 1
2320      2431 1
2321      2432 1
2322      2433 1
2323      2434 1
2324      2435 1
2325      2436 1
2326      2437 1
2327      2438 1
2328      2439 1
2329      2440 1
2330      2441 1
2331      2442 1
2332      2443 1
2333      2444 1
2334      2445 1
2335      2446 1
2336      2447 1
2337      2448 1
2338      2449 1
2339      2450 1
2340      2451 1
2341      2452 1
2342      2453 1
2343      2454 1
2344      2455 1
2345      2456 1
2346      2457 1
2347      2458 1
2348      2459 1
2349      2460 1
2350      2461 1
2351      2462 1
2352      2463 1
2353      2464 1
2354      2465 1
2355      2466 1
2356      2467 1
2357      2468 1
2358      2469 1
2359      2470 1
2360      2471 1
2361      2472 1
2362      2473 1
2363      2474 1
2364      2475 1
2365      2476 1
2366      2477 1
2367      2478 1
2368      2479 1
2369      2480 1
2370      2481 1
2371      2482 1
2372      2483 1
2373      2484 1
2374      2485 1
2375      2486 1
2376      2487 1
2377      2488 1
2378      2489 1
2379      2490 1
2380      2491 1
2381      2492 1
2382      2493 1
2383      2494 1
2384      2495 1
2385      2496 1
2386      2497 1
2387      2498 1
2388      2499 1
2389      2500 1
2390      2501 1
2391      2502 1
2392      2503 1
2393      2504 1
2394      2505 1
2395      2506 1
2396      2507 1
2397      2508 1
2398      2509 1
2399      2510 1
2400      2511 1
2401      2512 1
2402      2513 1
2403      2514 1
2404      2515 1
2405      2516 1
2406      2517 1
2407      2518 1
2408      2519 1
2409      2520 1
2410      2521 1
2411      2522 1
2412      2523 1
2413      2524 1
2414      2525 1
2415      2526 1
2416      2527 1
2417      2528 1
2418      2529 1
2419      2530 1
2420      2531 1
2421      2532 1
2422      2533 1
2423      2534 1
2424      2535 1
2425      2536 1
2426      2537 1
2427      2538 1
2428      2539 1
2429      2540 1
2430      2541 1
2431      2542 1
2432      2543 1
2433      2544 1
2434      2545 1
2435      2546 1
2436      2547 1
2437      2548 1
2438      2549 1
2439      2550 1
2440      2551 1
2441      2552 1
2442      2553 1
2443      2554 1
2444      2555 1
2445      2556 1
2446      2557 1
2447      2558 1
2448      2559 1
2449      2560 1
2450      2561 1
2451      2562 1
2452      2563 1
2453      2564 1
2454      2565 1
2455      2566 1
2456      2567 1
2457      2568 1
2458      2569 1
2459      2570 1
2460      2571 1
2461      2572 1
2462      2573 1
2463      2574 1
2464      2575 1
2465      2576 1
2466      2577 1
2467      2578 1
2468      2579 1
2469      2580 1
2470      2581 1
2471      2582 1
2472      2583 1
2473      2584 1
2474      2585 1
2475      2586 1
2476      2587 1
2477      2588 1
2478      2589 1
2479      2590 1
2480      2591 1
2481      2592 1
2482      2593 1
2483      2594 1
2484      2595 1
2485      2596 1
2486      2597 1
2487      2598 1
2488      2599 1
2489      2600 1
2490      2601 1
2491      2602 1
2492      2603 1
2493      2604 1
2494      2605 1
2495      2606 1
2496      2607 1
2497      2608 1
2498      2609 1
2499      2610 1
2500      2611 1
2501      2612 1
2502      2613 1
2503      2614 1
2504      2615 1
2505      2616 1
2506      2617 1
2507      2618 1
2508      2619 1
2509      2620 1
2510      2621 1
2511      2622 1
2512      2623 1
2513      2624 1
2514      2625 1
2515      2626 1
2516      2627 1
2517      2628 1
2518      2629 1
2519      2630 1
2520      2631 1
2521      2632 1
2522      2633 1
2523      2634 1
2524      2635 1
2525      2636 1
2526      2637 1
2527      2638 1
2528      2639 1
2529      2640 1
2530      2641 1
2531      2642 1
2532      2643 1
2533      2644 1
2534      2645 1
2535      2646 1
2536      2647 1
2537      2648 1
2538      2649 1
2539      2650 1
2540      2651 1
2541      2652 1
2542      2653 1
2543      2654 1
2544      2655 1
2545      2656 1
2546      2657 1
2547      2658 1
2548      2659 1
2549      2660 1
2550      2661 1
2551      2662 1
2552      2663 1
2553      2664 1
2554      2665 1
2555      2666 1
2556      2667 1
2557      2668 1
2558      2669 1
2559      2670 1
2560      2671 1
2561      2672 1
2562      2673 1
2563      2674 1
2564      2675 1
2565      2676 1
2566      2677 1
2567      2678 1
2568      2679 1
2569      2680 1
2570      2681 1
2571      2682 1
2572      2683 1
2573      2684 1
2574      2685 1
2575      2686 1
2576      2687 1
2577      2688 1
2578      2689 1
2579      2690 1
2580      2691 1
2581      2692 1
2582      2693 1
2583      2694 1
2584      2695 1
2585      2696 1
2586      2697 1
2587      2698 1
2588      2699 1
2589      2700 1
2590      2701 1
2591      2702 1
2592      2703 1
2593      2704 1
2594      2705 1
2595      2706 1
2596      2707 1
2597      2708 1
2598      2709 1
2599      2710 1
2600      2711 1
2601      2712 1
2602      2713 1
2603      2714 1
2604      2715 1
2605      2716 1
2606      2717 1
2607      2718 1
2608      2719 1
2609      2720 1
2610      2721 1
2611      2722 1
2612      2723 1
2613      2724 1
2614      2725 1
2615      2726 1
2616      2727 1
2617      2728 1
2618      2729 1
2619      2730 1
2620      2731 1
2621      2732 1
2622      2733 1
2623      2734 1
2624      2735 1
2625      2736 1
2626      2737 1
2627      2738 1
2628      2739 1
2629      2740 1
2630      2741 1
2631      2742 1
2632      2743 1
```

00000000G	00	04	SE DD 00005	PUSHL SP
	08	05	AC DD 00007	PUSHL EXIT FLAG
00000000G	00	00	02 FB 0000A	CALLS #2. DBG\$NCIS_REMOVE
		50	E8 00011	BLBS R0, 1\$
		BE	FA 00014	CALLG @MESSAGE_VECT, LIB\$SIGNAL
		04	0001C 1\$:	RET

: 2221

: 2230

: 2232

: Routine Size: 29 bytes,    Routine Base: DBG\$CODE + 0BE1

```
2123      2233 1 GLOBAL ROUTINE dbg$cis_add (pointer, length, type,
2124          2234 1           repeat_count, while_clause): NOVALUE =
2125          2235 1
2126          2236 1     ++
2127          2237 1     FUNCTIONAL DESCRIPTION:
2128          2238 1     Adds a link to the command input stream.
2129          2239 1     Note - this routine now just calls the routine DBGSNCIS_ADD in
2130          2240 1     the module DBGNEXTCE.
2131          2241 1
2132          2242 1     FORMAL PARAMETERS:
2133          2243 1     pointer - The address of either a buffer or a RAB to be placed
2134          2244 1           in the dsc$sa pointer field of the new link.
2135          2245 1     length - The length of the above buffer. (0 for RAB)
2136          2246 1     type - The type of the link to be added
2137          2247 1     repeat_count - For a link of type "doloop" [(created during processing
2138          2248 1           of REPEAT N TIMES ( . . . ) command], this represents the
2139          2249 1           number of remaining iterations.
2140          2250 1     while_clause - For a link of type "while", this points to a counted
2141          2251 1           ascii string with the while clause.
2142          2252 1
2143          2253 1     IMPLICIT INPUTS:
2144          2254 1           The head of the command input stream
2145          2255 1
2146          2256 1     IMPLICIT OUTPUTS:
2147          2257 1           None
2148          2258 1
2149          2259 1     ROUTINE VALUE:
2150          2260 1           None
2151          2261 1
2152          2262 1     SIDE EFFECTS:
2153          2263 1           None
2154          2264 1
2155          2265 2
2156          2266 2
2157          2267 2     BEGIN
2158          2268 2     LOCAL
2159          2269 2     message_vect; ! Holds message argument vector.
2160          2270 2
2161          2271 2     ! DBGSNCIS_ADD will return 'success' (1) if all goes well.
2162          2272 2
2163          2273 2     IF NOT dbg$ncis_add (.pointer, .length, .type,
2164          2274 2           .repeat_count, .while_clause, 0,
2165          2275 2           message_vect)
2166          2276 2
2167          2277 2
2168          2278 2
2169          2279 2
2170          2280 2
2171          2281 2
2172          2282 2
2173          2283 2
2174          2284 1     THEN
2175
2176          2275 3     BEGIN
2177          2276 3           ! Set up to signal error.
2178          2277 3
2179          2278 3
2180          2279 3     EXTERNAL ROUTINE
2181          2280 3           lib$signal : ADDRESSING_MODE (GENERAL);
2182          2281 3
2183          2282 3     BUILTIN
2184          2283 2           callg;
2185          2284 1           callg (.message_vect, lib$signal);
2186
2187          2285 2
2188
2189          2286 1     END;
2190
2191          2287 1
2192
2193          2288 1
2194
2195          2289 1
2196
2197          2290 1
2198
2199          2291 1
2200
2201          2292 1
2202
2203          2293 1
2204
2205          2294 1
2206
2207          2295 1
2208
2209          2296 1
2210
2211          2297 1
2212
2213          2298 1
2214
2215          2299 1
2216
2217          2300 1
2218
2219          2301 1
2220
2221          2302 1
2222
2223          2303 1
2224
2225          2304 1
2226
2227          2305 1
2228
2229          2306 1
2230
2231          2307 1
2232
2233          2308 1
2234
2235          2309 1
2236
2237          2310 1
2238
2239          2311 1
2240
2241          2312 1
2242
2243          2313 1
2244
2245          2314 1
2246
2247          2315 1
2248
2249          2316 1
2250
2251          2317 1
2252
2253          2318 1
2254
2255          2319 1
2256
2257          2320 1
2258
2259          2321 1
2260
2261          2322 1
2262
2263          2323 1
2264
2265          2324 1
2266
2267          2325 1
2268
2269          2326 1
2270
2271          2327 1
2272
2273          2328 1
2274
2275          2329 1
2276
2277          2330 1
2278
2279          2331 1
2280
2281          2332 1
2282
2283          2333 1
2284
2285          2334 1
2286
2287          2335 1
2288
2289          2336 1
2290
2291          2337 1
2292
2293          2338 1
2294
2295          2339 1
2296
2297          2340 1
2298
2299          2341 1
2300
2301          2342 1
2302
2303          2343 1
2304
2305          2344 1
2306
2307          2345 1
2308
2309          2346 1
2310
2311          2347 1
2312
2313          2348 1
2314
2315          2349 1
2316
2317          2350 1
2318
2319          2351 1
2320
2321          2352 1
2322
2323          2353 1
2324
2325          2354 1
2326
2327          2355 1
2328
2329          2356 1
2330
2331          2357 1
2332
2333          2358 1
2334
2335          2359 1
2336
2337          2360 1
2338
2339          2361 1
2340
2341          2362 1
2342
2343          2363 1
2344
2345          2364 1
2346
2347          2365 1
2348
2349          2366 1
2350
2351          2367 1
2352
2353          2368 1
2354
2355          2369 1
2356
2357          2370 1
2358
2359          2371 1
2360
2361          2372 1
2362
2363          2373 1
2364
2365          2374 1
2366
2367          2375 1
2368
2369          2376 1
2370
2371          2377 1
2372
2373          2378 1
2374
2375          2379 1
2376
2377          2380 1
2378
2379          2381 1
2380
2381          2382 1
2382
2383          2383 1
2384
2385          2384 1
2386
2387          2385 1
2388
2389          2386 1
2390
2391          2387 1
2392
2393          2388 1
2394
2395          2389 1
2396
2397          2390 1
2398
2399          2391 1
2400
2401          2392 1
2402
2403          2393 1
2404
2405          2394 1
2406
2407          2395 1
2408
2409          2396 1
2410
2411          2397 1
2412
2413          2398 1
2414
2415          2399 1
2416
2417          2400 1
2418
2419          2401 1
2420
2421          2402 1
2422
2423          2403 1
2424
2425          2404 1
2426
2427          2405 1
2428
2429          2406 1
2430
2431          2407 1
2432
2433          2408 1
2434
2435          2409 1
2436
2437          2410 1
2438
2439          2411 1
2440
2441          2412 1
2442
2443          2413 1
2444
2445          2414 1
2446
2447          2415 1
2448
2449          2416 1
2450
2451          2417 1
2452
2453          2418 1
2454
2455          2419 1
2456
2457          2420 1
2458
2459          2421 1
2460
2461          2422 1
2462
2463          2423 1
2464
2465          2424 1
2466
2467          2425 1
2468
2469          2426 1
2470
2471          2427 1
2472
2473          2428 1
2474
2475          2429 1
2476
2477          2430 1
2478
2479          2431 1
2480
2481          2432 1
2482
2483          2433 1
2484
2485          2434 1
2486
2487          2435 1
2488
2489          2436 1
2490
2491          2437 1
2492
2493          2438 1
2494
2495          2439 1
2496
2497          2440 1
2498
2499          2441 1
2500
2501          2442 1
2502
2503          2443 1
2504
2505          2444 1
2506
2507          2445 1
2508
2509          2446 1
2510
2511          2447 1
2512
2513          2448 1
2514
2515          2449 1
2516
2517          2450 1
2518
2519          2451 1
2520
2521          2452 1
2522
2523          2453 1
2524
2525          2454 1
2526
2527          2455 1
2528
2529          2456 1
2530
2531          2457 1
2532
2533          2458 1
2534
2535          2459 1
2536
2537          2460 1
2538
2539          2461 1
2540
2541          2462 1
2542
2543          2463 1
2544
2545          2464 1
2546
2547          2465 1
2548
2549          2466 1
2550
2551          2467 1
2552
2553          2468 1
2554
2555          2469 1
2556
2557          2470 1
2558
2559          2471 1
2560
2561          2472 1
2562
2563          2473 1
2564
2565          2474 1
2566
2567          2475 1
2568
2569          2476 1
2570
2571          2477 1
2572
2573          2478 1
2574
2575          2479 1
2576
2577          2480 1
2578
2579          2481 1
2580
2581          2482 1
2582
2583          2483 1
2584
2585          2484 1
2586
2587          2485 1
2588
2589          2486 1
2590
2591          2487 1
2592
2593          2488 1
2594
2595          2489 1
2596
2597          2490 1
2598
2599          2491 1
2600
2601          2492 1
2602
2603          2493 1
2604
2605          2494 1
2606
2607          2495 1
2608
2609          2496 1
2610
2611          2497 1
2612
2613          2498 1
2614
2615          2499 1
2616
2617          2500 1
2618
2619          2501 1
2620
2621          2502 1
2622
2623          2503 1
2624
2625          2504 1
2626
2627          2505 1
2628
2629          2506 1
2630
2631          2507 1
2632
2633          2508 1
2634
2635          2509 1
2636
2637          2510 1
2638
2639          2511 1
2640
2641          2512 1
2642
2643          2513 1
2644
2645          2514 1
2646
2647          2515 1
2648
2649          2516 1
2650
2651          2517 1
2652
2653          2518 1
2654
2655          2519 1
2656
2657          2520 1
2658
2659          2521 1
2660
2661          2522 1
2662
2663          2523 1
2664
2665          2524 1
2666
2667          2525 1
2668
2669          2526 1
2670
2671          2527 1
2672
2673          2528 1
2674
2675          2529 1
2676
2677          2530 1
2678
2679          2531 1
2680
2681          2532 1
2682
2683          2533 1
2684
2685          2534 1
2686
2687          2535 1
2688
2689          2536 1
2690
2691          2537 1
2692
2693          2538 1
2694
2695          2539 1
2696
2697          2540 1
2698
2699          2541 1
2700
2701          2542 1
2702
2703          2543 1
2704
2705          2544 1
2706
2707          2545 1
2708
2709          2546 1
2710
2711          2547 1
2712
2713          2548 1
2714
2715          2549 1
2716
2717          2550 1
2718
2719          2551 1
2720
2721          2552 1
2722
2723          2553 1
2724
2725          2554 1
2726
2727          2555 1
2728
2729          2556 1
2730
2731          2557 1
2732
2733          2558 1
2734
2735          2559 1
2736
2737          2560 1
2738
2739          2561 1
2740
2741          2562 1
2742
2743          2563 1
2744
2745          2564 1
2746
2747          2565 1
2748
2749          2566 1
2750
2751          2567 1
2752
2753          2568 1
2754
2755          2569 1
2756
2757          2570 1
2758
2759          2571 1
2760
2761          2572 1
2762
2763          2573 1
2764
2765          2574 1
2766
2767          2575 1
2768
2769          2576 1
2770
2771          2577 1
2772
2773          2578 1
2774
2775          2579 1
2776
2777          2580 1
2778
2779          2581 1
2780
2781          2582 1
2782
2783          2583 1
2784
2785          2584 1
2786
2787          2585 1
2788
2789          2586 1
2790
2791          2587 1
2792
2793          2588 1
2794
2795          2589 1
2796
2797          2590 1
2798
2799          2591 1
2800
2801          2592 1
2802
2803          2593 1
2804
2805          2594 1
2806
2807          2595 1
2808
2809          2596 1
2810
2811          2597 1
2812
2813          2598 1
2814
2815          2599 1
2816
2817          2600 1
2818
2819          2601 1
2820
2821          2602 1
2822
2823          2603 1
2824
2825          2604 1
2826
2827          2605 1
2828
2829          2606 1
2830
2831          2607 1
2832
2833          2608 1
2834
2835          2609 1
2836
2837          2610 1
2838
2839          2611 1
2840
2841          2612 1
2842
2843          2613 1
2844
2845          2614 1
2846
2847          2615 1
2848
2849          2616 1
2850
2851          2617 1
2852
2853          2618 1
2854
2855          2619 1
2856
2857          2620 1
2858
2859          2621 1
2860
2861          2622 1
2862
2863          2623 1
2864
2865          2624 1
2866
2867          2625 1
2868
2869          2626 1
2870
2871          2627 1
2872
2873          2628 1
2874
2875          2629 1
2876
2877          2630 1
2878
2879          2631 1
2880
2881          2632 1
2882
2883          2633 1
2884
2885          2634 1
2886
2887          2635 1
2888
2889          2636 1
2890
2891          2637 1
2892
2893          2638 1
2894
2895          2639 1
2896
2897          264
```

SE		0000 00000	ENTRY	DBGSCIS_ADD, Save nothing	: 2233
		04 C2 00002	SUBL2	#4, SP	
		2E DD 00005	PUSHL	SP	: 2271
		7E D4 00007	CLRL	-(SP)	
7E	10	AC 7D 00009	MOVQ	REPEAT_COUNT -(SP)	: 2272
7E	08	AC 7D 0000D	MOVQ	LENGTH -(SP)	: 2271
00000000G	00	04 AC DD 00011	PUSHL	POINTER	
00000000G	08	07 FB 00014	CALLS	#7, DBG3NCIS_ADD	
00000000G	00	50 E8 0001B	BLBS	R0 1\$	: 2282
		BE FA 0001E	CALLG	MESSAGE_VECT, LIBSSIGNAL	
		04 00026 1\$:	RET		: 2284

; Routine Size: 39 bytes, Routine Base: DBGSCODE + 0BFE

```

: 2176      2285 1 MACRO
: 2177      M 2286 1 IF_SIGNAL (code) =
: 2178      M 2287 1   IF .signal_flag NEQ 0
: 2179      M 2288 1     THEN
: 2180      M 2289 1       BEGIN
: 2181      M 2290 1         IF NOT
: 2182      M 2291 1           ( IF %LENGTH GTR 1
: 2183      M 2292 1             THEN dbg$snout_info (code, %REMAINING)
: 2184      M 2293 1             ELSE dbg$snout_info (code))
: 2185      M 2294 1         THEN
: 2186      M 2295 1           BEGIN
: 2187      M 2296 1             .signal_flag = (IF %LENGTH GTR 1
: 2188      M 2297 1               THEN
: 2189      M 2298 1                 dbg$make_arg_vect (code, %REMAINING)
: 2190      M 2299 1               ELSE
: 2191      M 2300 1                 dbg$make_arg_vect (code));
: 2192      M 2301 1           RETURN sts$k_severe;
: 2193      M 2302 1
: 2194      M 2303 1       END
: 2195      M 2304 1     ELSE
: 2196      M 2305 1       BEGIN
: 2197      M 2306 1         IF %LENGTH GTR 1
: 2198      M 2307 1           THEN
: 2199      M 2308 1             SIGNAL (code, %REMAINING)
: 2200      M 2309 1           ELSE
: 2201      M 2310 1             SIGNAL (code)
: 2202      M 2311 1       END %;
: 2203      M 2312 1
: 2204      M 2313 1 MACRO
: 2205      M 2314 1   SET_FLAG (param_num) =
: 2206      M 2315 1     LOCAL
: 2207      M 2316 1       signal_flag;
: 2208      M 2317 1
: 2209      M 2318 1     signal_flag = (IF actualcount () GTR param_num
: 2210      M 2319 1           THEN
: 2211      M 2320 1             actualparameter (actualcount())
: 2212      M 2321 1           ELSE
: 2213      M 2322 1             0) %;
: 2214      M 2323 1
: 2215      M 2324 1   END
: 2216      M 2325 0 ELUDOM

```

.EXTRN LIB\$SIGNAL

#### PSECT SUMMARY

Name	Bytes	Attributes
DBG\$PLIT	571	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$CODE	3109	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$OWN	32	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2)

## Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	86	0	1000	00:01.9
\$255\$DUA28:[DEBUG.OBJ]STRUDEF.L32;1	32	0	0	7	00:00.1
\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	124	8	97	00:01.9
\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	11	2	31	00:00.3
\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	36	9	22	00:00.3
\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	30	20	12	00:00.3

## COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGLEVEL1/OBJ=OBJ\$:DBGLEVEL1 MSRC\$:DBGLEVEL1/UPDATE=(ENH\$:DBGLEVEL1)

: Size: 3109 code + 603 data bytes  
: Run Time: 01:03.9  
: Elapsed Time: 03:10.8  
: Lines/CPU Min: 2181  
: Lexemes/CPU-Min: 18423  
: Memory Used: 304 pages  
: Compilation Complete

0084 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

DBGIFTHEN  
LIS

DBGLANVEC  
LIS

DBGGEN  
LIS

DBGLANGOP  
LIS

DBGLEVEL  
LIS

0085 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

DBGLEVEL3  
LIS

DBGLIB  
LIS